LA RIVISTA DI ELETTRONICA APPLICATA, TECNOLOGIE E PRODOTTI

Fare ELETTRONICA

N° 233 - NOVEMBRE 2004 - ANNO 20

€ 4,50 - Frs 9,00

HARDWARE

- GENERATORE DI FUNZIONI PROGRAMMABILE
- GENERATORE PER ARGENTO COLLOIDALE
- GE.CO.AS. GENERATORE DI CODICE ASSEMBLER PER MICROCONTROLLORI PIC
- GLI STABILIZZATORI DI TENSIONE: I REGOLATORI DI TIPO SWITCHING. I DC-DC CONVERTER

PRATICAMENTE

- OSCILLATORI AL QUARZO: CONTASECONDI

TECNOLOGIE SPERIMENTALI

- APPLICAZIONI EMBEDDED: IL PROTOCOLLO MODBUS
- PROGETTIAMO UN RAZZO: ALTIMETRO BAROMETRICO A DOPPIA ESPULSIONE

TUTORIAL

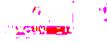
- SMARTCARD: LA SIM: UN ESEMPIO DI SMARTCARD A MICROPROCESSORE
- VITAMINA C: STRUTTURA E LEGGIBILITÁ DEL CODICE

COSTO ZERO

- CONVERTITORE PER LAMPADE FLUORESCENTI
- CARICABATTERIE NI-CD/MH A 3 PORTATE CON LM 317

ROBOMANIA

- FEBOT: IL FIRMWARE DELLA SCHEDA MADRE
- ASPIRONE: NON PERDE LA BUSSOLA
- INTRODUZIONE ALLA ROBOTICA: CAPACITÁ SENSORIALE

NOTEPAD

SMARTCARD

Come funziona e come gestire la SIM del vostro telefonino

USIAMO LE CPLD

Generatore di funzioni programmabile

FEBOT

Il firmware della scheda madre

MEDICINA NATURALE

Generatore per argento colloidale

CONVERTITORE

per lampade fluorescenti

CARICABATTERIE NI-CD/MH

a 3 portate con LM 317

ISSN 1591-2272

40233

9 771591 227008

Poste Italiane Spa - Spedizione in abbonamento Postale - D.L. 353/2003 (conv. In L. 27/02/2004 n. 46) art. 1, comma1, DCB Milano. In caso di mancato recapito, restituire all'editore che si impegna a pagare la relativa tassa presso il CPM di Roserio - Milano

www.farelettronica.com

editoriale

WWW.FARELETTRONICA.COM

Sin dalla sua prima comparsa sulla Rete il sito di Fare Elettronica ha suscitato un grande successo ed interesse. Dopo un periodo di revisione (durato un po' troppo per la verità) è tornato online con rinnovato vigore qualche settimana dopo il mio arrivo in redazione (due anni or sono).
Oggi, dopo alcuni cambiamenti di minore entità, siamo finalmente pronti con una nuovissima versione che troverete on line in contemporanea con l'uscita di questo numero: rinnovato il layout, introdotti numerosi servizi (sono troppi per elencarli tutti), rinnovato il forum (molto utilizzato e quindi inadeguato nella precedente versione), aggiunta a grande richiesta una chat.
Questi sono solamente quattro dei temi più importanti che hanno caratterizzato il nostro lavoro. Attendiamo molti commenti al riguardo e, nel prossimo numero, parleremo in un apposito articolo di tutte le nuove funzionalità introdotte.
Il sito web è molto importante per noi, poiché moltissimi lettori amano contattarci attraverso internet ed utilizzare il sito per accedere alle risorse a cui, sempre più spesso, nella rivista si fa riferimento; è il valore aggiunto, quella "marcia in più" per noi fondamentale per darvi sempre il massimo possibile, utilizzando le tecnologie più fruibili e disponibili a basso costo.
Il nuovo sito avrà anche un rinnovato "shop", un mercatino con prodotti molto interessanti a prezzi molto contenuti. Ogni abbonato potrà acquistare utilizzando un buono sconto (il meglio definito "coupon") e toccare con mano quanto sia facile e conveniente.

Ma le novità, cari lettori, non si fermano al solo sito, abbiamo pronto il piano editoriale per il 2005. Moltissime le novità e tutte di grande rilievo: oltre al già annunciato corso di elettronica analogica (Elettronicando), dal prossimo numero parte un corso teorico/pratico sugli alimentatori switching (pagina 101), stiamo preparando un corso sulle CPLD, una serie di articoli dedicati alla costruzione di strumenti di laboratorio, un corso sul sistema DCC e sull'intelligenza artificiale.
Ma vi ho detto troppo e rischio di rovinarvi la sorpresa, molto altro bolle in pentola, quindi vi consiglio di non perdere i prossimi numeri!

Il numero che state per leggere è come ogni mese ricco di contenuti di alta qualità: apre *Gianroberto Negri* con la terza parte dedicata al mondo embedded in cui illustra un protocollo molto importante: il *MODBUS*; sul fronte delle CPLD, argomento sul quale *Agostino Rolando* ha grande competenza, presentiamo un bellissimo *Generatore di funzioni* che andrà ad arricchire il vostro laboratorio; per gli amanti della medicina naturale *Andrea Marani* vi guida alla costruzione di un *Generatore di argento colloidale*.
Ritorna dopo due mesi di assenza *Eugenio Cosolo* con un nuovo strumento necessario per la costruzione del missile, un *Altimetro barometrico*; Marco Lento propone due semplici progetti dedicati a chi è alle prime armi mentre *Dario Mazzeo* accontenta tutti gli appassionati di microcontrollori PIC con un interessante *generatore di codice assembler*.
Il *FEbot*, che sta riscuotendo tanto interesse, passa ad un nuovo livello, infatti in questa puntata presentiamo la prima parte relativa al codice per far funzionare il microcontrollore utilizzato (PICmicro); ma non è il solo articolo dedicato alla robotica, infatti *Marco Fabbri* aggiunge la bussola ad *Aspirone* il robot aspirapolvere presentato qualche numero fa e *Massimiliano Bracci* continua il suo tutorial sulla robotica parlando della *capacità sensoriale*.
Chiudono il numero le nuove puntate di *Vitamina C* e *Gli stabilizzatori di tensione* scritti rispettivamente da *Antonio di Stefano* e *Nico Grilloni* e, per finire, una nuova puntata di *Praticamente* questo mese dedicata agli *oscillatori al quarzo*.

Vi lascio alla scoperta di questo interessantissimo numero, vi auguro una piacevole lettura e vi invito in edicola a Dicembre con un numero davvero speciale.

Tiziano Galizia
t.galizia@farelettronica.com

www.farelettronica.com

DIRETTORE RESPONSABILE:
GianCarmelo Moroni

DIRETTORE DI REDAZIONE:
Tiziano Galizia (t.galizia@farelettronica.com)

PROGETTO GRAFICO E IMPAGINAZIONE:
Graficonsult - Milano (info@graficonsult.com)

HANNO COLLABORATO:
Maurizio Del Corso, Andrea Marani, Marco Lento, Marco Fabbri, Dario Mazzeo, Esteban Mascarella, Eugenio Cosolo, Massimiliano Bracci, Gianroberto Negri, Antonio Di Stefano, Nico Grilloni, Andrea Marani, Agostino Rolando, Giuseppe Modugno,

DIREZIONE - REDAZIONE - PUBBLICITÁ
INWARE srl
Via Cadorna, 27/31 - 20032 Cormano (MI)
Tel. 02.66504794 - 02.66504755 - Fax 02.66508225
info@inware.it - www.inwaredizioni.it

STAMPA:
ROTO 2000
Via Leonardo da Vinci, 18/20 - 20080 Casarile (MI)

DISTRIBUZIONE:
Parrini & C. S.p.a.
Viale Forlanini, 23 - 20134 Milano.
Il periodico Fare Elettronica è in attesa del numero di iscrizione al ROC

UFFICIO ABBONAMENTI
PARRINI & C. S.p.a. Servizio abbonamenti
Viale Forlanini, 23 - 20134 Milano
Per informazioni, sottoscrizione o rinnovo dell'abbonamento:
Telefono: 02.66504794 - Fax: 02.66508225
Email: abbonamenti@farelettronica.com

Poste Italiane Spa - Spedizione in abbonamento Postale - D.L. 353/2003 (conv. In L. 27/02/2004 n. 46) art. 1, comma1, DCB Milano
Abbonamento per l'Italia: € 39,00
Abbonamento per l'estero: € 99,00
Per la sottoscrizione degli abbonamenti, utilizzare il modulo stampato in ultima pagina.
Gli arretrati potranno essere richiesti, per iscritto, al seguente costo:
Numero singolo: € 7,50
Numero doppio: € 9,00

Autorizzazione alla pubblicazione del Tribunale di Milano n. 647 del 17/11/2003 INWARE srl.

Richieste di assistenza

Per richiedere assistenza o chiarimenti sugli articoli pubblicati, vi preghiamo di contattare l'autore, il cui nome ed indirizzo email è sempre riportato sotto il titolo dell'articolo stesso.
Nel caso ciò non fosse possibile potete scrivere a mailbox@farelettronica.com, ricordandovi di specificare il numero della rivista ed il titolo dell'articolo per il quale chiedete chiarimenti, oltre al vostro nome, cognome ed indirizzo email.
Tutte le richieste con informazioni insufficienti o anonime non saranno prese in considerazione.

Collaborare con Fare Elettronica

La redazione di Fare Elettronica è alla ricerca di collaboratori per la stesura di articoli, progetti, tutorials, rubriche e libri.
Le richieste di collaborazione vanno indirizzate a Tiziano Galizia (t.galizia@farelettronica.com) e accompagnate, se possibile, da una breve descrizione delle vostre competenze tecniche e/o editoriali, oltre che da un elenco degli argomenti e/o progetti che desiderate proporre.

Come contattarci

Indirizzo email della Redazione:
redazione@farelettronica.com
Indirizzo email dell'Ufficio Abbonamenti:
abbonamenti@farelettronica.com
I nostri numeri telefonici:
Telefono 02.66504794 Fax 02.66508225
Il nostro indirizzo postale:
INWARE Edizioni
Via Cadorna, 27/31
20032 Cormano (MI)

Elenco inserzionisti

Alterlogix	61
Artek	25-55
Blu Nautilus	31
Elettroshop	67-105
Eurocom	53
Futura	11-79
G.P.E. kit	89
Grifo	II cop
Idea Elettronica	17
Ital Electronics	47
Parsic	39-65
Pianeta Elettronica	83-109
Pianeta Musica	35
Scuola RadioElettra	IV cop

SOMMARIO

hardware

Generatore di funzioni programmabile 32
di Agostino Rolando

Generatore per argento colloidale 42
di Andrea Marani

GE.CO.AS. Generatore di codice assembler per microcontrollori PIC 80
di Dario Mazzeo

Gli stabilizzatori di tensione (sesta parte): I regolatori di tipo switching. I DC-DC converter 84
di Nico Grilloni

praticamente

Oscillatori al quarzo: Contasecondi 74
di Maurizio Del Corso

tecnologie sperimentali

Applicazioni embedded (terza parte): Il protocollo MODBUS 14
di Gianroberto Negri

Progettiamo un razzo (nona parte): Altimetro barometrico a doppia espulsione 66
di Eugenio Cosolo e Esteban Mascarella

tutorial

Smartcard (sesta parte): La sim: un esempio di smartcard a microprocessore 54
di Giuseppe Modugno

Vitamina C (tredicesima parte): Struttura e leggibilità del codice 48
di Antonio Di Stefano

costo zero

Convertitore per lampade fluorescenti 90
di Marco Lento

Caricabatterie NI-CD/MH a 3 portate con LM 317 92
di Marco Lento

robomania

Febot (terza parte): Il firmware della scheda madre 94
di Maurizio Del Corso e Tiziano Galizia

Aspirone (terza parte): Non perde la bussola 102
di Marco Fabbri

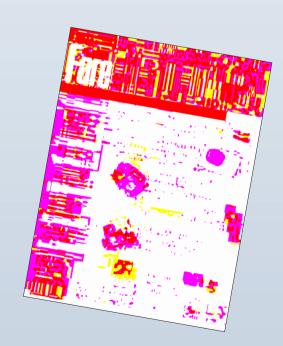
Introduzione alla robotica (quarta parte): Capacità sensoriale 106
di Massimiliano Bracci

rubriche

Mailbox 6

News 8

Notepad 12

Le fiere e mostre mercato di Novembre 2004 72

mailbox

Dubbi, perplessità, malfunzionamenti, opinioni, commenti o richieste?

Inviateli a: **mailbox@farelettronica.com**

Oppure scriveta a:
Mailbox - Redazione di Fare Elettronica
Inware srl
Via Cadorna, 27/31 - 20032 Cormano (MI)

Le lettere più interessanti saranno pubblicate in queste pagine. Per quanto possibile, inoltre, cercheremo di dare una risposta privata a chiunque ci scriverà via email.

D DALLA TEORIA ALLA PRATICA

Seguendo gli articoli sugli stabilizzatori di tensione, ho progettato un alimentatore del quale vi invio lo schema elettrico (figura 2 n.d.r.). Ho fatto tutti i calcoli per avere 5V in uscita, ma in realtà ne misuro solo 3,xx. Ho eseguito i calcoli più volte, ma i risultati mi sembrano corretti. Dove ho sbagliato?

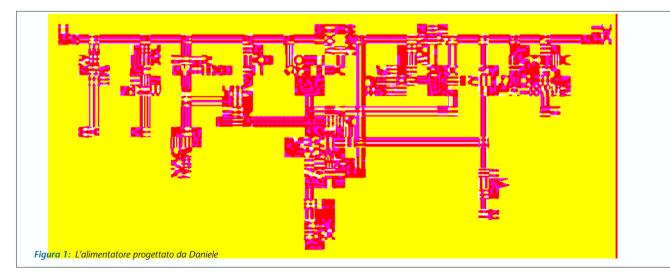
Daniele Lombardo

R Seguendo gli articoli pubblicati, la tensione di uscita del circuito è data dalla relazione:

Vout = (Vd1+Vd2+Vd3+Vbe2)(1+R1/R2)

Dove Vd1, Vd2 e Vd3 sono le tensioni di soglia dei rispettivi diodi e Vbe2 la tensione tra base ed emettitore di Q2.

Sicuramente nei tuoi calcoli avrai assunto sicuramente 0,7V come tensione ai capi di ciascun diodo. In realtà quando un diodo è in conduzione non è detto che la tensione ai suoi capi sia precisamente 0,7V, ma dipende dal tipo di diodo utilizzato e due diodi identici possono avere tensioni leggermente diverse. Normalmente i valori della tensione di soglia per un diodo varia da 0,2V (nei diodi Schottky) fino a 1,2V (nei LED). La soluzione è sostituire i tre diodi in serie con un diodo Zener da 2V collegando l'anodo alla massa. In alternativa è possibile scegliere tre diodi che abbiano una tensione di soglia vicina agli 0,7V, oppure usare un trimmer al posto di R1 in modo da poter variare leggermente il guadagno del sistema ed ottenere i 5V in uscita.

Figura 1: *L'alimentatore progettato da Daniele*

PROBLEMI CON IL CD "ANNATA 2003"

Ho acquistato recentemente il CD-ROM Fare Elettronica "Annata 2003", ma quando tento aprire il pdf di una rivista il mio PC si blocca. Ho un Pentium 166MHz con 64Mb di RAM. Potete aiutarmi?

Maurizio Vincenti

I file PDF delle riviste contenuti nel CDROM "Annata 2003", sono stati realizzati utilizzando una elevata risoluzione al fine di ottenere delle stampe di ottima qualità, per questo motivo le dimensioni di tali file sono di circa 50-60Mb. Aprire un file di queste dimensioni con un PC avente un quantitativo di memoria RAM non troppo elevato (come nel caso del lettore) provoca un rallentamento delle prestazioni del PC talvolta assimilabile ad un blocco totale del sistema. In realtà il sistema non è bloccato, ma sta eseguendo un largo numero di operazioni di "swapping" (mappatura di pagine di memoria su hard disk) per poter aprire il file. Consigliamo quindi almeno 128Mb di RAM e di utilizzare versioni di Acrobat Reader aggiornate anche se i pdf contenuti nel CDROM sono stati testati con la versione 6.1 presente nel CD stesso.
Il problema segnalatoci dal lettore, non si presenta con il CDROM "PIC microcontroller By Example" disponibile già da settembre, in quanto in questo caso i file pdf sono di dimensioni minori contenendo ciascuno una singola lezione del corso.

DIODO VARICAP

Spettabile redazione, ho sentito parlare di diodi varicap. Vorrei sapere cosa sono e come funzionano.
Grazie e complimenti per la rivista che seguo sempre con molta attenzione.

Gianfranco Chiarini

Un diodo VARICAP è un diodo che ha la proprietà di offrire una capacità diversa a seconda della tensione inversa applicata ai suoi capi. Il suo comportamento è quindi assimilabile ad un condensatore il cui valore capacitivo è controllabile da una tensione. Un diodo varicap (VARIable CAPacitor) va utilizzato in polarizzazione inversa ovvero con la tensione sul catodo maggiore di quella sull'anodo. Il suo funzionamento si basa sulla modulazione della zona di svuotamento in prossimità della giunzione interna. La zona di svuotamento è una zona in cui non ci sono cariche libere, ma solo ioni fissi quindi la struttura è molto simile ad un condensatore a facce piane e parallele. Aumentando la tensione inversa, la zona di svuotamento si allarga, quindi la capacità diminuisce. Questo effetto è tipico di tutti i diodi, ma per i VARICAP è molto più accentuato. Nella figura seguente è riportato il simbolo elettrico del diodo varicap.

Figura 2: Simbolo elettrico del diodo Varicap

OSCILLATORE

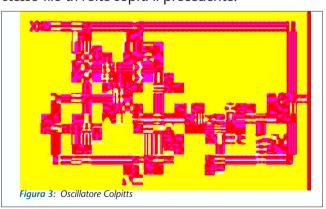
Per eseguire l'allineamento di una coppia di walkie-talkie avrei bisogno di un oscillatore in banda cittadina che possa coprire i 27MHz che sia economico e velocemente realizzabile. Grazie per la vostra eventuale risposta.

Marco Celi

È possibile utilizzare il circuito di figura 3 che è un oscillatore Colpitts in cui il transistor lavora con base a massa attraverso un condensatore da 4,7nF. La sintonia grossa avviene regolando il nucleo della bobina, mentre quella fine è stabilita dalla capacità dei diodi varicap la quale dipende, a sua volta, dalla tensione ad essi applicata dal trimmer. Il segnale generato si sviluppa ai capi della bobina il cui secondario lo mette a disposizione con una impedenza di 100 Ohm. Il transistor è al germanio per frequenze elevate (può essere usato anche un AF239), mentre il primario della bobina è formato da 9 spire di filo da 0,25mm avvolte su un supporto da 6mm con nucleo. Il secondario è un link da due spire dello stesso filo avvolte sopra il precedente.

Figura 3: Oscillatore Colpitts

NEWS

423

*Questo spazio è gentilmente offerto da **EONews**, il Quindicinale di notizie e commenti per l'industria elettronica di **VNU Business Publications Italia**.*

IDEE DI PROGETTO: LA NUOVA INIZIATIVA SUL WEB DI ELETTRONICA OGGI

"Idee di progetto – Design Ideas" è la nuova rubrica di Elettronica Oggi che è stata lanciata dal mese di Settembre 2003. Caratteristica saliente di questa nuova iniziativa è che sarà completamente ed esclusivamente on line ed accessibile dal sito www.ilb2b.it.
L'obbiettivo principale è creare una vera e propria libreria di idee alla quale tutti coloro che operano in maniera professionale nel mondo dell'elettronica possano "catturare" informazioni e suggerimenti utili per il loro lavoro quotidiano. Questo nuovo strumento vi permette di scaricare, con un solo click, tutte le risorse necessarie per risolvere velocemente qualsiasi problema e, in ultima analisi, minimizzare il time to market.

DRIVE PER MOTORI PASSO-PASSO

ELCAM, distributore esclusivo per il territorio italiano degli azionamenti Copley Controls, introduce sul mercato StepNet, il drive per motori passo-passo con interfaccia CANopen che può essere implementato nelle reti CANopen fino ad oggi riservate ai soli servoazionamenti.
StepNet offre alte performances nel controllo dei motori passo-passo dando inoltre la possibilità di scegliere il case più adatto all'applicazione (card, panel, module).
StepNet può essere utilizzato per applicazioni multiasse su rete CANopen (DSP-402 protocol); l'intelligenza integrata gli permette di eseguire profili point to point con curve ad "S", profili complessi, con interpolazioni polinomiali, dando informazioni di posizione, velocità e tempo.
StepNet utilizza un controllo ad orientamento di campo per massimizzare le prestazioni velocità/coppia del motore. Un circuito di damping permette lo smorzamento delle oscillazioni e riduce i tempi di assestamento della velocità. Il drive pilota il motore a passo intero, mezzo passo e micropassi per incrementare la risoluzione e diminuire il ripple di velocità.

***readerservice.it** 423 - 53*

HIP OTTICI

Intense ha in corso la fabbricazione di chip che integrano fino a 100 o più componenti ottici multifunzione su una singola piastrina.
Questo notevole successo di integrazione dimostra gli alti rendimenti conseguibili con l'esclusivo processo Qwl di Intense, per mezzo del quale è possibile rendere disponibile una piattaforma per costruire affidabilmente chip ottici in grado di fornire considerevoli riduzioni in termini di dimensioni e costi in applicazioni quali networking ottico, stampa e difesa.

***readerservice.it** 423 - 54*

DIODI DIFFUSI AL PLATINO

International Rectifier ha introdotto i diodi epitassiali a 600 V diffusi al platino e a ripristino veloce - 60Apu06 e 60Epu06.
Il processo di diffusione del platino controlla accuratamente il tempo di vita dei portatori di carica.
La capacità di stabilire o regolare le caratteristiche (velocità) di ripristino del diodo permette di ottimizzare prestazioni ed il rendimento. Le applicazioni per i nuovi diodi comprendono il raddrizzamento di uscita negli alimentatori industriali a commutazione. Ad esempio, sono molto adatti alla realizzazione di saldatrici ad alta frequenza, in alternata e Tig con corrente continua in uscita da 80 a 200 A, dove, a causa della presenza di picchi di tensione, viene raccomandato l'utilizzo di dispositivi a 600 V.

***readerservice.it** 423 - 55*

RELÈ DI POTENZA PER PCB

Il nuovo relè di potenza serie Rx per Pcb di Tyco Electronics è realizzato con contatti senza cadmio, conformi alle direttive RoHs.
Ha le stesse portate di contatto, specifiche di bobina e parametri di isolamento dei tradizionali relè di potenza per Pcb alti 25,5 mm, come la serie Schrack Rp di Tyco Electronics stessa, ma si presenta in un alloggiamento di basso profilo, 15,7 mm, come i relè di nuova generazione. Disponibile con "cover" opaco (bianco) oppure trasparente, ha la piedinatura standard di 3,5 e 5 mm. Proposto in versioni unipolari da 6 o 12 A, bipolare 8 A e bobina in Dc e in Ac, temperatura di funzionamento compresa tra -40 e +85 gradi C (+70 gradi C per le versioni bipolari in Ac), è approvato Ul e Vde.

***readerservice.it** 423 - 57*

IC CONTROLLER

Isl6227 è il nome di un nuovo Ic controller Pwm doppio, progettato da Intersil per alimentare memorie Ddr, schede grafiche, chipset ed altri sistemi nei Pc notebook. La sua ampia gamma di tensioni d'ingresso permette la conversione diretta della tensione da un adattatore Ac/Cc, da un accumulatore al litio-ione o da un bus di sistema a 5 o 3,3 V. Si propone come un dispositivo migliorativo a piedinatura compatibile per Isl6225.

readerservice.it 423 - 60

MICROCONTROLLORI RISC

Toshiba ha annunciato l'introduzione di una nuova famiglia di microcontrollori Risc a 32 bit e a chip singolo a tecnologia Mips. I nuovi dispositivi Tx19A70 offrono contemporaneamente il controllo vettoriale per motori azionati da inverter e il controllo applicativo specialistico per condizionatori d'aria, lavatrici e impianti di refrigerazione. È caratterizzata da un nucleo di elaborazione Tx19A Risc a 32 bit e ad alte prestazioni di recente sviluppo ed include sia l'architettura Misp32 Isa a 32 bit per la velocità e l'Ase di Mips16e, un set di istruzioni ad alta efficienza di codice e che contiene istruzioni aggiuntive di Toshiba.
La maggior parte delle istruzioni può essere eseguita in un solo ciclo di 17,8 ns.

readerservice.it 423 - 61

MOSFET DI POTENZA

Finalizzato ad interruttori di batterie, amplificatori di potenza, interruttori di carico e applicazioni di ricarica, il nuovo Mosfet di potenza a canale p chipscale Micro Foot Si8413Db di Siliconix (Silverstar-Celdis) eroga una resistenza di on max di appena 48 mohm ad un gate drive di 4,5 V, con tensione di rottura di -20 V. Misurante appena 1,54 x 1,54 x 0,62 mm, offre prestazioni comparabili a quelli di dispositivi racchiusi in package Tsop-6 occupando un quarto dello spazio.
Se utilizzato in Pda, telefoni cellulari ed altri prodotti elettronici portatili, consente di rendere più piccoli e sottili i prodotti finali, per aggiungervi più funzioni, e/o per estenderne i tempi di esecuzione tra ricariche di batterie.

readerservice.it 423 - 63

CONTROLLER PWM

Texas Instruments ha presentato un innovativo circuito integrato di gestione della potenza per convertitori a bassa tensione di uscita e ad alta densità.
Siglato Ucc2540, concerne un nuovo controller Pwm sincrono, lato secondario, che semplifica il progetto di applicazioni con alimentazioni di uscita multiple, quali moduli di comunicazione dati e telecom, alimentatori industriali, computer, strumentazione medicale e di prova e alimentatori mercantili.
In grado di supportare un intervallo di tensioni di alimentazione in ingresso da 2,7 a 35 V, può funzionare da un bus 3,3 V e pilota due Mosfet a canale N in sincronia ad una frequenza di commutazione fino a 1 MHz per erogare correnti elevate ad alti rendimenti, minimizzando nel contempo le dimensioni di induttori e condensatori.

readerservice.it 423 - 64

IC TRANSPONDER

Em Microelectronics (Ebv Elektronik) propone una famiglia di circuiti integrati Rfid che funziona a 13,56 MHz per sistemi di identificazione senza contatto.La famiglia è composta dai dispositivi: Em4035, Em-4135, Em4034 e Em4094 ed è realizzata con la tecnologia Rfid da 13,56 MHz e dimostra l'impegno dell'azienda nel coprire tutte le frequenze Rfid al fine di offrire la massima flessibilità. Aderente alla norma Iso15693, è in grado di leggere 20-40 etichette/s.

readerservice.it 423 - 65

MOLTIPLICATORE VETTORIALE

Analog Devices ha ampliato la sua famiglia di Ic Rf ad alte prestazioni con il moltiplicatore vettoriale a due canali Adl5390 per applicazioni di controllo di fasi e guadagni. In grado di sostituire fino a sei componenti discreti in precedenza necessari per eseguire controlli di fasi e guadagni in apparecchiature di infrastrutture wireless, è particolarmente adatto per connessioni in banda larga nell'ultimo miglio quali Lmds, Wll e apparecchiature di infrastrutture cellulari per Gsm, W-Cdma e Cdma2000.

readerservice.it 423 - 67

POWERMOSFET

Nec Electronics ha commercializzato una nuova serie di PowerMosfet realizzata con l'avanzata tecnologia Umos-4.
La nota serie Np di Nec è stata ampliata con lo sviluppo di 15 nuovi prodotti, puntando al mercato dei dispositivi a bassa tensione (30, 40 e 55 V) quali il settore automotive. Questi nuovi prodotti sono disponibili in package compatibili con i popolari To-252 e To-263 e sono caratterizzati da resistenze

estremamente basse, alte correnti e stabilità in temperatura fino a 175 gradi C. La caratteristica principale è la resistenza di conduzione Rdsn(on) ultrabassa, fino a 1,7 mohm max (a 30 V, To-263Zp).

readerservice.it 423 - 68

SWITCH USB

Fairchild Semiconductor ha annunciato l'immediata disponibilità del suo nuovo Fsusb11, un doppio switch Usb Spdt, a larghezza di banda elevata, a bassa potenza, fornente multiplessaggi/demultiplessaggi a rapidi tempi di commutazione di segnali Usb 1.1 e audio analogici. Progettato specificatamente per applicazioni Usb, è ideale per applicazioni di commutazione in telefoni cellulari ultraportatili, computer e periferiche, prodotti elettronici di consumo ed altri dispositivi dotati di porte Usb 1.1.
Fornisce prestazioni superiori tramite basso consumo di potenza, eccezionale protezione Esd e bassa distorsione delle armoniche totali.
È alloggiato in un package Micropak che misura appena 1,6 x 2,1 mm.

readerservice.it 423 - 70

AMPLIFICATORE PER LARGA BANDA

National Semiconductor presenta gli amplificatori high-speed LMH6738 e LMH6739, entrambi creati utilizzando la tecnologia di processo proprietaria di National VIP10, sono destinati alle applicazioni che richiedono una larga banda passante e una bassa distorsione. Con una banda passante a piccolo segnale di 750MHz questi dispositivi ad alte prestazioni trovano applicazione nel pilotaggio di segnali video ad alta risoluzione nei proiettori LCD, nelle applicazioni multimediali, nello switching e routing dei segnali video, per conference room e per i sistemi HDTV.

readerservice.it 423 - 71

PROCESSORE D'APPLICAZIONE

Sh-Mobile3 (Sh73180) è il prossimo modello ad alte prestazioni della serie di processori d'applicazione Sh-Mobile per sistemi di telefonia mobile annunciato da Renesas Technology. Incorpora il nuovo core Cpu Risc Sh4Al-Dsp a 32 bit, che offre prestazioni superiori di circa 2,3 volte rispetto ai precedenti modelli.
Integra, inoltre, un motore per la grafica bidimensionale e tridimensionale, un acceleratore hardware full Mpeg-4 e supporto per modulo fotocamera da 3 megapixel.
Queste caratteristiche permettono di eseguire svariate tipologie di applicazioni multimediali avanzate per i telefoni cellulari di prossima generazione di alta gamma.

readerservice.it 423 - 73

MEMORIA SDRAM

La Sdram 32 M x 72 di White Electronic Designs è una memoria Dram, Cmos ad alta velocità utilizzante 5 chip contenenti 536.870.912 bit. Ciascun chip è configurato internamente come una Dram a banchi quadrupli con interfaccia singola. Presenta frequenze di clock di 125 e 100 MHz ed utilizza architetture canalizzate interne per ottenere funzionamenti ad alta velocità e banchi interni per nascondere precariche e accessi a file, consentendo cambi di indirizzi di colonne per ogni ciclo di clock. Lunghezze di burst programmabili comprendono 1, 2, 4, 8 o pagine complete. Offre 8.192 cicli di refresh e include modalità di autorefresh e refresh automatici e precarica automatica. È racchiusa in un package Pbga, 32 x 25 mm, e pesa 2,5 g.

readerservice.it 423 - 74

MEMORIA FLASH

La memoria Flash 3V 64 Mbyte (W78M64V-Xsbx) di White Electronic Designs è stata progettata per complementare processori e controllori di memorie ad alte prestazioni.
È configurabile dall'utente come 8 M x 64, 2 banchi di 8 M x 32 o 4 banchi di 8 M x 16 ed offre risparmi di spazio del 31% in confronto con densità equivalenti in package chip scale, diminuzioni di I/O del 50% e conteggi di parti ridotti. Presenta tempi di accesso di 70, 90, 100 e 120 ns.
Funzionalità Read/Write simultanee consentono otto letture continue da un banco eseguendo contemporaneamente funzioni erase/program in un altro, con commutazione a latenza zero da operazioni di lettura a quelle di scrittura. Quattro banchi distinti permettono fino a 4 operazioni simultanee/dispositivo.
Le opzioni di I/O disponibili comprendono 1,8 e 3 V.

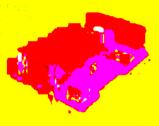
readerservice.it 423 - 76

COME OTTENERE MAGGIORI INFORMAZIONI

EONews offre il servizio "reader service" che vi consente, utilizzando l'apposito codice riportato alla fine di ogni news, di ricevere maggiori informazioni.

Visitate il sito **www.readerservice.it** e compilate la cartolina virtuale con i vostri dati, il numero della rivista, questo mese il **423**, ed i numeri di reader service presi dalle notizie che vi interessa approfondire.

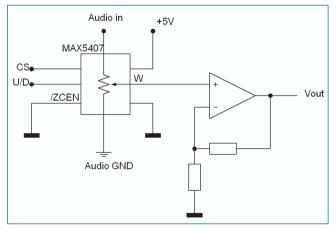
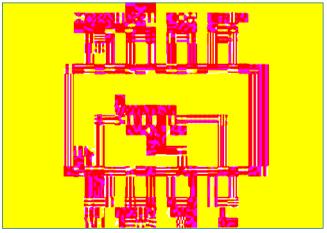
EONEWS provvederà, tempestivamente, a contattare le aziende interessate, che invieranno al vostro indirizzo tutta la documentazione disponibile.

Network-enable

notepad

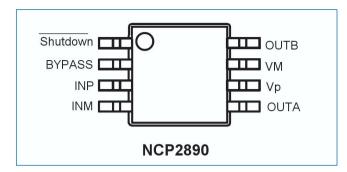
Dal blocco note di Fare Elettronica una raccolta di idee da tenere sempre a portata di mano

Questa rubrica ha lo scopo di fornire degli schemi applicativi o idee di progetto dei componenti elettronici più interessanti, selezionati per voi dalla redazione. Tutti gli schemi presentati sono elaborazioni di quelli ufficiali proposti dai produttori nella documentazione ufficiale.

CONTROLLO ELETTRONICO DI VOLUME CON MAX5407 24

Il MAX5407 è un potenziometro digitale ideale per applicazioni audio. Una volta abilitato dal pin CS, è possibile scegliere l'incremento o il decremento della resistenza mediante il pin U/D.

La funzionalità AERO CROSSING abilitabile mediante il pin ZCEN permette di variare la resistenza solo quando la tensione ai capi del potenziometro è zero: in questo modo si riduce drasticamente la distorsione di segnali audio. In figura una applicazione per il controllo del volume.

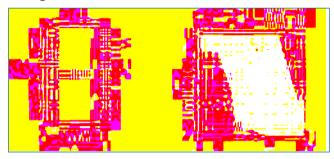
MINI AMPLIFICATORE DI POTENZA DA 1W CON NCP2890 25

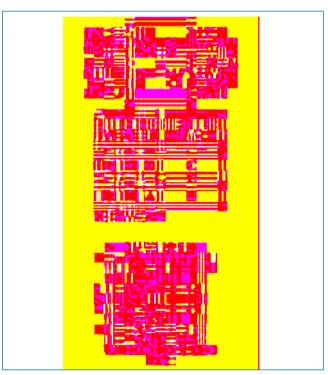
NCP2890, un amplificatore di potenza particolarmente adatto ad applicazioni miniaturizzate viste le dimensioni ridottissime ed i minimi componenti esterni richiesti per il corretto funzionamento. È in grado di fornire una potenza di 1W su un carico di 8 Ohm se alimentato a 5V, oppure, 320mW su 4 Ohm con alimentazione a 2,6V.

RS232 IN WIRELESS BLUETOOTH CON IL MODULO HCS-100 26

HCS-100 è un modulo Bluetooth che consente il trasferimento di segnali RS232 via radio secondo lo standard Bluetooth. Dispone di tutti i segnali RS232 in TTL che invia via radio attraverso l'antenna integrata. In figura una tipica applicazione per la remotizzazione di un pannello operatore su un PLC eliminando così il cavo seriale.

FILTRO ANTI-ALIASING DIGITALE PROGRAMMABILE 27

LTC1564 è un filtro digitale anti-aliasing con frequenza di taglio e guadagno programmabili. Può

essere alimentato da 2,7V a 10V. Come mostrato in figura i limiti di programmabilità vanno da un guadagno unitario (0dB) e frequenza di taglio di 10KHz fino ad un guadagno di 25dB e frequenza di taglio di 150KHz.

MAX4734: MULTIPLEXER ANALOGICO A 4 CANALI 28

Il MAX4734 è un multiplexer analogico a 4 canali prodotto da Maxim. Consente di riportare in uscita il segnale analogico di uno dei quattro ingressi a seconda della combinazione dei due ingressi di controllo. È alimentabile con tensione singola da 1.6V a 3,6V ed assorbe una potenza inferiore a 4mW se alimentato a 3V. Gli ingressi di controllo sono da 1,8V CMOS compatibili e la resistenza di contatto è dell'ordine di 0.8W. In figura è riportato il pinout, la tabella di verità ed il grafico della resistenza di contatto al variare della tensione di uscita VCOM e della tensione di alimentazione V+.

APPLICAZIONI EMBEDDED
terza parte

IL PROTOCOLLO MODBUS

Gianroberto Negri
info@gnrs.it

Prima di vedere l'implementazione vera e propria dell'applicazione in Visual Basic, vedremo in questa puntata il protocollo ModBus ed introdurremo anche i concetti di formato dati del tipo Big Indian e Little Indian. Nelle puntate successive ne implementeremo un sottoinsieme denominato ASCII che, verrà incapsulato nei messaggi UDP che, verranno scambiati tra l'emulatore PLC (emula solo dal punto di vista del comportamento e non riguardo all'implementazione del 6113-3) ed il sistema di controllo e supervisione.

IL PROTOCOLLO MODBUS

Uno dei protocolli più diffusi in ambito industriale è quello MODBUS. Vi sono sostanzialmente due implementazioni:

- Quella che fa capo ad un collegamento o BUS seriale RS232/RS485.
- Quella che fa capo ad un collegamento o BUS di rete TCP/IP od UPD.

In questo articolo ci occuperemo soltanto di quello su BUS Seriale e delle sue due implementazioni, quella RTU e quella ASCII.

Regole generali

Entrambe le implementazioni sono del tipo Master/Slave. Solo un Master nel medesimo tempo può essere connesso ed effettuare una richiesta che transiterà sul BUS. Più Slave sino ad un massimo di 247 possono essere invece connessi al medesimo BUS e rispondere alle richieste fatte dal Master. Solo il Master può iniziare una comunicazione, quindi gli Slave non possono in maniera arbitraria attivare una comunicazione. Oltretutto nessuno Slave può comunicare con un altro Slave. Come già detto solo una transazione alla volta può transitare sul BUS.

Il Master effettua le richieste agli Salve in due modi:

UNICAST

- In esso il Master indirizza un singolo Slave. Appena ha ricevuto e processato la richiesta del Master, lo Slave ritorna quanto richiesto.
- Sostanzialmente ogni transazione consiste in due messaggi o telegrammi, cioè richiesta da parte del master a cui corrisponde una risposta da parte dello Salve.
- Nel medesimo momento soltanto uno Salve può essere indirizzato tra 1 e 247 possibili.

BROADCAST

- In esso il Master invia una richiesta / telegramma a tutti gli slave presenti sul BUS. Non viene effettuata alcuna risposta da parte degli Slave. Si tratta essenzialmente di una richiesta di scrittura da parte del master da far eseguire a tutti gli Slave presenti.
- L'indirizzo utilizzato è 0.

Regole di indirizzamento

Gli indirizzi utilizzabili sono 256 e sono così suddivisi:

- 0 Broadcast.
- 1 – 247 Indirizzi individuali Slave.
- 248 - 255 Riservati.

L'indirizzo 0 come già detto è riservato al Brodcast ed è riconosciuto da tutti gli Slave. Il Master non ha uno specifico indirizzo, solo gli Slave lo hanno e lo stesso è unico per tutto il BUS seriale Modbus.

Descrizione del frame

Il ModBus Application protocol definisce un livello base dello stesso denominato PDU dall'Inglese Protocol Data Unit che, è indipendente dal tipo di BUS utilizzato per la trasmissione. Questo vuol dire che esso è valido sia per il BUS seriale che TCP/IP. Il PDU è composto da due campi:

Figura 1: Struttura del ModBus PDU

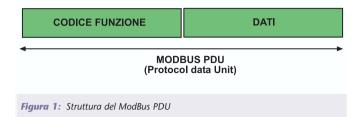
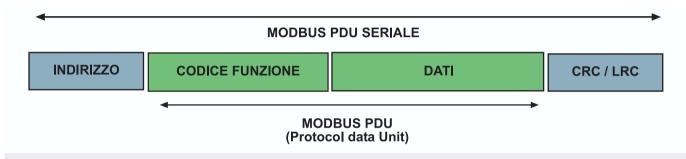



Figura 2: Struttura di un telegramma per il BUS Seriale

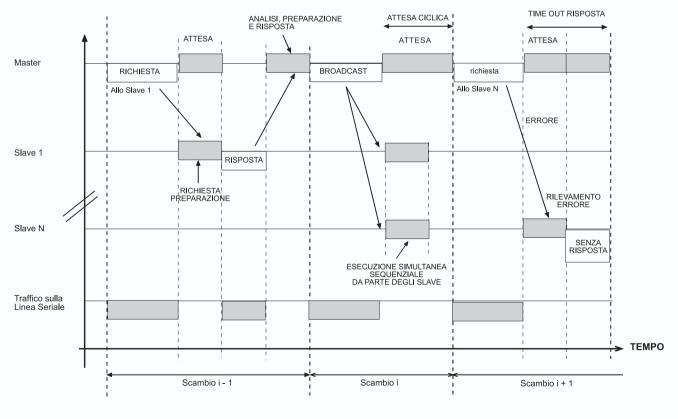



Figura 3: Diagramma Temporale; Master/Slave

- Codice Funzione
- Dati

La mappatura del protocollo nell'ambito del BUS seriale aggiunge alcuni campi al PDU prima visto e lo stesso diventa:
- Indirizzo
- Codice Funzione
- Dati
- CRC/LRC

Vediamoli in dettaglio. Iniziamo dal campo Indirizzo che, serve come fa comprendere il nome ad indirizzare in maniera univoca uno degli Slave collegati al BUS.
Il campo Codice Funzione trasmette allo Slave indirizzato una precisa richiesta. Ad esempio un comando di scrittura di una Word o di uno o più Bit. Ad esso segue una area dati in cui sono contenuto i parametri della funzione trasmessa, nel caso di una richiesta da parte del Master oppure i dati richiesti o la conferma dell'avvenuta esecuzione della funzione trasmessa, nel caso di uno Slave.
L'ultimo campo contiene il risultato di un calcolo ridondante di controllo effettuato sul contenuto del messaggio. L'algoritmo utilizzato è il CRC per l'RTU e L'LRC PER L'ASCII.

In figura 3 possiamo osservare un diagramma temporale raffigurante 3 tipici esempi di comunicazione Master/Slave.

Nota: la durata delle fasi delle RICHIESE, delle RISPOSTE e del BROADCAST, dipende dalle caratteristiche della comunicazione (Lunghezza del frame, velocità della linea...).
La durata delle fasi delle ATTESE e delle PREPARAZIONI dipende dal tempo che, lo Slave impiega per elaborale.

IMPLEMENTAZIONE MODBUS RTU

Analizziamo ora il ModBus RTU. Iniziamo dal significato di RTU: Remote Terminal Unit.
In esso ogni BYTE composto notoriamente da 8 Bit, contiene due caratteri Esadecimali, ciascuno composto da 4 Bit, Nibble in gergo informatico.

Indirizzo Slave	Codice Funzione	Dati	CRC	CRC
1 Byte	1 Byte	Da 0 a 252 Bytes	1 Byte CRC LOW	1 Byte CRC HI

Figura 4: *Struttura di un messaggio/telegramma RTU*

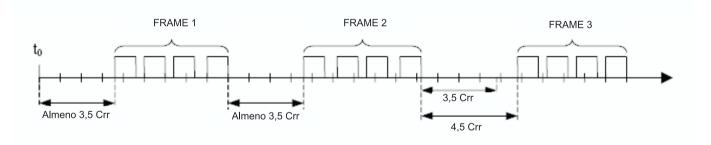



Figura 5: *Tempificazioni tra i messaggi RTU*

Messaggio / Telegramma MODBUS

Inizio	Indirizzo Slave	Codice Funzione	Dati	CRC	CRC	Fine
>= 3,5 crr	1 Byte	1 Byte	N x 8 Bits	1 Byte CRC LOW	1 Byte CRC HI	>= 3,5 crr

Figura 6: *Tempificazioni tra i messaggi RTU con evidenziato il messaggio stesso*

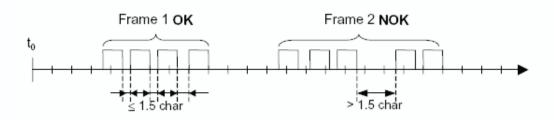



Figura 7: *Tempificazioni tra caratteri nell'ambito di un messaggio*

È una forma di trasmissione molto compatta e veloce rispetto a quella ASCII. In essa ogni Messaggio/Telegramma viene trasmesso inserito in un flusso continuo di caratteri.
Per determinare l'inizio o la fine di un messaggio è stato convenuto che deve passare un tempo uguale o maggiore a 3,5 caratteri e che nell'ambito della trasmissione di ogni carattere del messaggio non trascorra più tempo di 1,5 caratteri. Traducendoli in millisecondi in una trasmissione a 19.200 Bps 3,5 caratteri equivalgono a 1,750 ms, mentre 1,5 caratteri equivalgono a 0,750 ms. Tali tempi valgono anche per velocità di trasmissioni maggiori di 19.200 Bps. Questo significa che tempificazioni più ridotte non vengono richieste anche se la velocità del BUS seriale aumenta notevolmente. Vediamo i concetti espressi in figura 5. In essa possiamo osservare orizzontalmente la linea del tempo ed all'inizio della stessa troviamo un intervallo di tempo di appunto 3,5 caratteri. Segue il messaggio ModBus

Messaggio / Telegramma MODBUS ASCII

Inizio	Indirizzo Slave	Codice Funzione	Dati	LRC	Fine
1 crr 1 Byte :	2 crr 2 Byte	2 crr 2 Byte	Da 0 a 2 x 252 crr	2 crr 2 Byte	2 crr 2 Byte CR , LF

Figura 8: Struttura di un messaggio/telegramma ASCII

RTU, poi un nuovo intervallo di 3,5 caratteri ed un altro messaggio. L'intervallo che, osserviamo dopo tale messaggio non è di 3,5 caratteri ma di 4,5 caratteri. Infatti per determinare l'inizio/fine di un messaggio vengono accettati valori anche superiori a 3,5 caratteri. Nella figura 6 per riassumere troviamo un tempo di 3,5 caratteri che identifica l'inizio del messaggio, il messaggio ed infine sempre un tempo di 3,5 caratteri che ne determina la fine.
Come già detto esistono delle precise tempificazioni anche nell'ambito del messaggio stesso.
Affinché il messaggio possa essere considerato valido tra la trasmissione di un carattere ed il successivo non può trascorrere un tempo maggiore di 1,5 caratteri. Questo viene ben evidenziato dalla figura 7 dove troviamo due messaggi di cui uno non valido (il secondo) a causa del fatto che tra un carattere ed il seguente il tempo trascorso è maggiore di 1,5 caratteri. Tale messaggio viene scartato, non considerandolo valido.

Poiché si rende necessario essere totalmente sicuri che il messaggio sia veramente corretto, oltre le tempificazioni è stato introdotto un ulteriore controllo il CRC.
Si tratta di una verifica di ridondanza ciclica (in Inglese cyclical redundancy checking o CRC). La stessa viene applicata all'intero messaggio ed il risultato viene posto negli ultimi 2 Bytes dello stesso nei campi CRC appunto.
Ecco come viene calcolato:

1. Caricare con il valore 0xFFFF (tutti 1) un registro a 16 bit (8+8) denominato CRC Register.
2. Fare lo XOR del (primo) byte del messaggio con il byte basso del CRC Register, e mettere il risultato ancora nel CRC Register.
3. Shiftare a destra di uno il CRC Register, caricando con 0 il bit più significativo (msb).
4. Valutare il bit uscito a destra nell'operazione di shift:
   a. Se vale 0, ripetere dal punto 3 (altro shift)
   b. Se vale 1, eseguire lo XOR del CRC Register col valore 0xA001
5. Ripetere i punti 3 e 4 fino ad effettuare 8 shift: a questo punto è stato elaborato il (primo) byte del messaggio.
6. Ripetere i punti da 2 a 5 per tutti i byte del messaggio.
7. Il contenuto finale del CRC Register rappresenta il valore del checksum.
8. Scambiare l'ordine dei due byte del CRC Register e accodarli al messaggio.

Nel punto 8 viene detto di scambiare i due Bytes, a tale proposito apriamo ora una parentesi su come i due Bytes ottenuti dal calcolo vengono memorizzati ed introduciamo i concetti di **Little Indian** e **Big Indian**.

Little Indian e **Big Indian** sono i termini che descrivono l'ordine con cui una macchina immagazzina i byte di una parola a 16 o 32 bits in memoria.

Big Indian (l'indiano più grande) è l'ordine per cui la parte più significativa viene scritta/memorizzata per prima, nel primo Byte o all'indirizzo più basso di memoria.

Little Indian (l'indiano più piccolo) è l'ordine per cui la parte meno significativa viene scritta/memo-

rizzata per prima, nel primo Byte o all'indirizzo più basso di memoria.

Vediamo un esempio:
Ipotizziamo che, abbiamo effettuato il calcolo del CRC richiesto dal ModBus RTU e che il risultato sia 1241h. Bene nell'ordine **Big Indian** i dati saranno scritti/memorizzati esattamente come sono, senza effettuare alcuno scambio. Per cui verrà scritto prima **12h** e poi **41h**.
Invece nell'ordine **Little Indian** i dati saranno scambiati, per cui verrà scritto/memorizzato prima **41h** e poi **12h**. Da quanto detto si comprende che il protocollo ModBus utilizza l'ordine **Little Indian** per scrivere/memorizzare i dati. Oltretutto questo formato e tipico dei Microprocessori Intel, mentre il **Big Indian** è tipico dei Microprocessori Motorola.
Nota: attenzione quindi quando scambiamo dati tra piattaforme differenti... Se lo scambio non funziona potrebbe essere dovuto al fatto che le stesse adottino un formato di dati differenti!

IMPLEMENTAZIONE MODBUS ASCII

Passiamo ora al ModBus ASCII che, sarà poi l'implementazione che adotteremo.

Contrariamente al ModBus RTU nell'ASCII non sono richieste tempificazioni strette che servono a determinare la reale dimensione e correttezza del messaggio. In esso l'inizio e la fine del messaggio è evidenziata da opportuni caratteri.
Infatti, mediante il carattere : (3Ah in esadecimale) ha inizio il messaggio e mediante i caratteri **CR** ed **LF** (0Dh e 0Ah in esadecimale) lo stesso termina.
Nel caso arrivi un messaggio che non inizia o termina con i caratteri previsti lo stesso sarà ignorato.

Rispetto il ModBus RTU esiste un ulteriore differenza: In ASCII occorre un numero doppio di Bytes per trasmettere lo stesso messaggio rispetto all'RTU. Quindi se per trasmettere l'indirizzo dello Slave interessato al messaggio, in RTU bastava un Byte in ASCII ne occorrono due. Questo per un ovvio motivo, il numero costituito da due cifre esadecimali che nell'RTU è contenuto in un unico Byte, in ASCII diviene due caratteri e quindi due Bytes.

Facciamo un esempio:

```
RTU: 41h Memorizzato in 1 Byte
ASCII: 4 e 1
HEX: 34h e 31h Memorizzati in 2 Bytes
```

Come per l'RTU anche per l'ASCII esiste un controllo necessario per essere totalmente sicuri che il messaggio sia veramente corretto. È meno rigido ma altrettanto affidabile la sua sigla è LRC che significa Longitudinal Redundancy Checking che, tradotto in Italiano diventa Controllo Ridondante Longitudinale. Vediamo come si calcola:

1. Sommare tutti I Bytes contenuti nel messaggio con esclusione del primo (:) e degli ultimi 2 CR/LF e naturalmente dei due Bytes in cui dovrà essere posto il risultato del calcolo LRC.
2. Sottrarre il valore esadecimale FFh (tutti I Bit a 1) dal risultato. Questo produce un complemento a 1.
3. Sommare 1 al risultato. Questo produce un complemento a 2.
4. Porre il primo carattere esadecimale (HI) nel primo Byte ed il secondo (LOW) nel secondo Byte.

Figura 9: *Suddivisione Codici Funzione ModBus*

				Codici Funzioni		
				Codice	Sottocodice	HEX
Accesso ai Dati	Accesso a Bit	Ingressi Fisico ai Singoli Bit	Leggi Ingressi Discreti	02		02
		Lettura/Scrittura Bits Interni o Fisici	Leggi Bits	01		01
			Scrivi Singolo Bit	05		05
			Scrivi Bit Multipli	15		0F
	Accesso a 16 Bits (World)	Ingresso Fisico ai Registri (World)	Leggi Registro Ingressi	04		04
		Lettura/Scrittura Registri (WORLD) Interni o Fisici	Leggi Registri Interni	03		03
			Scrivi Singolo Registro	06		06
			Scrivi Registri Multipli	16		10
			Leggi/Scrivi Registri Multipli	23		17
			Maschera Registro Di Scrittura	22		16
			Leggi Coda FIFO	24		18
	Accesso ai Records		Leggi Record File	20	6	14
			Scrivi Record File	21	6	15
Diagnostica			Legge lo Stato (Eccezione)	07		07
			Dignostica	08	00-18	
			Acquisisce il Contatore degli Eventi della Seriale	11		0B
			Acquisisce il Log degli Eventi della Seriale	12		0C
			Legge le Caratteristiche dello Slave ID	17		11

Figura 10: *Elenco Codici Funzione ModBus*

Codici funzione base

Vediamo alcuni Codici Funzione utilizzati per la maggiore nel protocollo ModBus. Essi permettono di accedere alle variabili contenute negli Slave.

Normalmente la scambio di dati nel protocollo avviene sotto forma di Word che, per intenderci sono variabili a 16 Bit composte da due Bytes posizionati come già detto secondo l'ordine Little Indian, con cioè la parte meno significativa scritta/memorizzata per prima, nel primo Byte o all'indirizzo più basso di memoria.

In figura 9 possiamo osservare le categorie dei Codici Funzione ModBus e vedere che sostanzialmente sono suddivise in 2 categorie, vediamole:

Codici Funzione Pubblici

- Sono predefini
- Garantiti unici
- Validati dall'organizzazione modbus.org
- Esiste documentazione pubblica
- Sottoposti a test di conformità
- Documentati in MB IETF RFC
- Include anche Codici Funzione non ancora utilizzati ma, riservati per futuri utilizzi
- Vanno dal Codice 1 al Codice 64, dal Codice 73 al 99 e dal Codice 111 al Codice 127

Codici Funzione definiti dall'Utilizzatore (Progettista del dispositivo ModBus)

- Vanno dal Codice 65 al 72 e dal Codice 100 al 110
- Possono essere implementati senza dover chie-

dere l'autorizzazione all'organizzazione modbus.org

- Rimane ovvio che lo stesso Codice Funzione può essere utilizzato da più di un costruttore e questo può portare all'esistenza di medesimi Codici Funzione per applicazioni differenti.

In figura 10 possiamo osservare l'intero elenco dei Codici Funzione utilizzati dal protocollo ModBus.

Una premessa prima di addentrarci nella spiegazione dei vari codici funzione, occorre evidenziare che i Bits che ad esempio vanno dal Bit 1 al Bit 16 in realtà vanno indirizzati dal Bit 0 al Bit 15. L'indirizzo iniziale parte sempre da 0 e non da 1. In informatica quando si ha a che fare con Bit, Byte ed indirizzi il numero zero occupa sempre una posizione e per la precisione, quella iniziale.

Vediamo il funzionamento di alcuni dei Codici Funzione presenti nell'elenco.

Codice Funzione 01 (0x01) Leggi Coils

Questa funzione viene utilizzata per leggere lo stato dei Bits contigui che vanno da 1 a 2000 simultaneamente, con una unica richiesta. Nel PDU che viene inviato allo Slave oltre ovviamente il Codice Funzione viene anche specificato l'indirizzo da cui iniziare la lettura e per quanti Bits contigui. Nella risposta lo Slave raggruppa i Bits in Bytes, divide cioè i Bits richiesti per 8 e li restituisce sotto forma di Bytes. Nell'esempio riportato nella parte bassa della figura 11 possiamo vedere che partendo dall'indirizzo esadecimale 00 13 (19 in decimale) che corrisponde al Bit numero 20, viene richiesto lo stato di 19 Bits consecutivi (00 13 in Hex). Dividendo 19 per 8 otteniamo 2 e rimangono 3 Bits che vengono posti in un terzo Byte ponendo a zero i restanti 5 Bit che lo compongono.

Possiamo anche osservare come i Bits vengono restituiti dallo Slave, nella forma cioè 27-20, 35-28,38-36.

Il Bit più alto rimane a sinistra, come di convenzione vengono normalmente rappresentati gli stessi.

Codice Funzione 02 (0x02) Leggi Ingressi Discreti (Fisici)

Funzione molto simile alla precedente, l'unica differenza e che, fa riferimento direttamente solo agli I/O Fisici e non eventualmente anche alla memoria.

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x01
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Coils	2 Bytes	Da 1 a 2000 (0x7D0)

Risposta

Codice Funzione	1 Byte	0x01
Numero Bytes Restituiti	1 Byte	N°
Stato Coils	n Bytes	n = N o N + 1

Errore

Codice Funzione	1 Byte	Codice Funzione + 0x80
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 01 - Leggi Coils

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	01	Codice Funzione	01
Indirizzo HI	00	Numero Bytes	03
Indirizzo LOW	13	Valore Stato Coils 27-20	CD
Numero di Coils HI	00	Valore Stato Coils 35-28	6B
Numero di Coils LOW	13	Valore Stato Colis 38-36	05

Figura 11: Codice Funzione 01 Leggi Coils

Codice Funzione 03 (0x03) Leggi dai Registri sino ad un massimo di 125 per volta

Questa funzione viene utilizzata per leggere il contenuto dei blocchi contigui dei registri interni. Il PDU specifica l'indirizzo di partenza ed il numero di registri contigui da leggere. Come sempre per il registro 1 si parte dall'indirizzo 0. La risposta da parte dello Slave è contenuta in un blocco a cui sono assegnati due Bytes per ogni singolo registro. Nel primo Byte è contenuta la parte alta HI del registro, mentre nel secondo Byte la parte bassa LOW. In Figura 13 possiamo vedere in basso un esempio di richiesta e relativa risposta da parte dello Slave.

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x02
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Ingressi	2 Bytes	Da 1 a 2000 (0x7D0)

Risposta

Codice Funzione	1 Byte	0x02
Numero Bytes Restituiti	1 Byte	N°
Stato Ingressi	N° x 1 Byte	

Errore

Codice Funzione	1 Byte	0x82
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 02 - Leggi Ingressi Discreti

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	02	Codice Funzione	02
Indirizzo HI	00	Numero Bytes	03
Indirizzo LOW	C4	Stato Ingressi 27-20	AC
Quantità Ingressi HI	00	Stato Ingressi 35-28	DB
Quantità Ingressi LOW	16	Stato Ingressi 38-36	35

Figura 12: Codice Funzione 02 Leggi Ingressi Discreti

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x03
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri	2 Bytes	Da 1 a 125 (0x7D)
Risposta		
Codice Funzione	1 Byte	0x03
Numero Bytes Restituiti	1 Byte	2 x N°
Valore Registri	N° x 2 Bytes	
Errore		
Codice Funzione	1 Byte	0x83
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 03 - Leggi Registri Holding

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	03	Codice Funzione	03
Indirizzo HI	00	Numero Bytes	06
Indirizzo LOW	6B	Valore Registro HI (108)	02
Numero di Registri HI	00	Valore Registro LOW (108)	2B
Numero di Registri LOW	03	Valore Registro HI (109)	00
		Valore Registro LOW (109)	00
		Valore Registro HI (110)	00
		Valore Registro LOW (110)	64

Figura 13: Codice Funzione 03 Leggi dai Registri sino ad un massimo di 125 per volta

Codice Funzione 04 (0x04) Leggi Registro Ingresso

Questa funzione viene utilizzata per leggere da 1 a 125 Registri di Input da uno Slave. Il PDU specifica l'indirizzo del registro di partenza ed il numero di registri da leggere. L'indirizzo parte da 0. I dati contenuti nella risposta sono contenuti in un blocco di 2 Bytes per ogni registro, in cui nel primo è contenuta la parte alta HI e nel secondo la parte Bassa LOW.

Codice Funzione 05 (0x05) Modifica/Scrive su di una Singola Uscita o Bit (Coils)

Questa funzione viene utilizzata per scrivere/modificare una singola uscita o Bit. La modifica dello stato ON/OFF dell'uscita avviene mediante la scrittura di una costante nel messaggio di richiesta verso lo Slave. La costante viene scritta in due Bytes. Per lo stato ON la costante da inserire nel primo e secondo Bytes è 0xFF 0x00, mentre per lo stato OFF è 0x00 e 0x00. Qualsiasi altro valore non ha effetto sullo stato dell'uscita. Il PDU specifica l'indirizzo del Coil da modificare. Gli indirizzi partono da 0. Lo Slave risponde restituendo il comando inviato dal Master.

Codice Funzione 06 (0x06) Scrivi Singolo Registro

Questa funzione viene utilizzata per scrivere/modificare un singolo registro. Il PDU specifica l'indirizzo ed il valore da scrivere nel Registro. Anche in questo caso gli indirizzi partono da 0. Lo Slave risponde con un eco del comando contenente il valore cambiato nello Slave.

Codice Funzione 07 (0x07) Leggi lo Stato (Eccezione) [Solo per Seriale]

Questa funzione viene utilizzata per leggere il contenuto degli 8 Bits di stato eccezione in un device remoto. Essa provvede un semplice metodo per accedere a tali informazioni non disponibili altrimenti. La normale risposta fornita dal device contiene lo stato di tali Bits. Vengono inviati entro un Byte.

Codice Funzione 08 – [00 -18] (0x08 – [0x00 – 0x12]) Diagnostica della linea seriale

Questa funzione provvede una serie di test per

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x04
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri di Ingresso	2 Bytes	Da 0x0001 a 0x007D
Risposta		
Codice Funzione	1 Byte	0x04
Numero Bytes Restituiti	1 Byte	2 x N°
Registri di Ingresso	N° x 2 Bytes	
Errore		
Codice Funzione	1 Byte	0x84
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 04 - Leggi Registri Ingressi

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	04	Codice Funzione	04
Indirizzo HI	00	Numero Bytes	02
Indirizzo LOW	08	Registro di Ingresso HI (9)	00
Quantità di Registri Ingresso HI	00	Registro di Ingresso LOW (9)	0A
Quantità di Registri Ingresso LOW	01		

Figura 14: Codice Funzione 04 Leggi Registri Ingresso

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x05
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Valore Uscita	2 Bytes	Da 0X0000 a 0xFF00
Risposta		
Codice Funzione	1 Byte	0x05
Numero Bytes Restituiti	2 Bytes	Da 0x0000 a 0xFFFF
Stato Ingtressi Richiesti	2 Bytes	Da 0X0000 a 0xFF00
Errore		
Codice Funzione	1 Byte	0x85
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 05 - Scrivi singolo Coil

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	05	Codice Funzione	05
Indirizzo HI	00	Indirizzo HI	00
Indirizzo LOW	AC	Indirizzo LOW	AC
Valore Uscita HI	FF	Valore Uscita HI	FF
Valore Uscita LOW	00	Valore Uscita LOW	00

Figura 15: Codice Funzione 05 Modifica/Scrive su di una Singola Uscita o Bit (Coils)

testare la comunicazione seriale tra il Master e lo Slave, permettendo anche di evidenziare eventuali errori interni allo Slave. Essa utilizza due codici funzione per definire il tipo di test da effettuare. Possono esserci molteplici cause che possono impedire la trasmissione tra il Master ed i vari Slave. Può anche capitare che uno Slave possa resettarsi se riceve un codice funzione che non è in grado di gestire. Questa funzione può essere utile per rimuovere od individuare delle situazioni di malfunzionamento.
Vediamo le varie opzioni:

00 Ritorna Query

I dati inviati allo Slave vengono restituiti per cui la richiesta da parte del master e la risposta dello Slave sono identiche. Si attiva una condizione di Loopback.

Sottofunzione	00 00
Dati Richiesta	Qualsiasi
Dati Risposta	Eco della richiesta

01 Ripartenza opzioni comunicazione

Il device remoto posto sulla linea seriale viene inizializzato e fatto ripartire. Tutti gli eventi inerenti alla comunicazione sono resettati. Funziona se non è stata attivato lo stato di Ascolto.
Quando il device remoto riceve la richiesta tenta un riavviamento ed esegue le relative prove di test previste dal costruttore dello Slave. Il valore FF 00 causa al ricevimento da parte dello Slave un Reset del LOG degli eventi della comunicazione. Il valore 00 00 genera invece un Riavvio.

Sottofunzione	00 01 - 00 00
Dati Richiesta	00 00 - FF 00
Dati Risposta	Eco della richiesta

02 Ritorna il registro di diagnostica

Il contenuto del registro diagnostico da 16 Bits viene ritornato dallo Slave come risposta.

Sottofunzione	00 02
Dati Richiesta	00 00
Dati Risposta	16 Bits Registro Diagnostica

03 Cambia il delimitatore ASCII

Serve per cambiare il limitatore LF nell'ambito del protocollo ASCII. Il codice ASCII che viene trasmesso allo Slave sostituirà LF.

Sottofunzione	00 03
Dati Richiesta	Carattere ASCII
Dati Risposta	Eco della richiesta

04 Forza in stato di ascolto

Serve per mettere uno Slave in stato di ascolto. In

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x06
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Valore Registro	2 Bytes	Da 0X0000 a 0xFF00
Risposta		
Codice Funzione	1 Byte	0x06
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Valore Registro	2 Bytes	Da 0X0000 a 0xFF00
Errore		
Codice Funzione	1 Byte	0x86
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 06 - Scrive Singolo Registro

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	06	Codice Funzione	06
Indirizzo HI	00	Indirizzo HI	00
Indirizzo LOW	01	Indirizzo LOW	01
Valore Registro HI	00	Valore Registro HI	00
Valore Registro LOW	03	Valore Registro LOW	03

Figura 16: *Codice Funzione 06 Scrivi/Modifica singolo Registro*

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x07
Risposta		
Codice Funzione	1 Byte	0x07
Numero Bytes Restituiti	1 Byte	Da 0x00 a 0xFF
Errore		
Codice Funzione	1 Byte	0x87
Codice Eccezione	1 Byte	01/04

ESEMPIO: Funzione 07 - Leggi Stato Eccezione (Solo Linea Seriale)

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	07	Codice Funzione	07
		Dati Risposta	6D

Figura 17: *Codice Funzione 07 Legge lo stato della Seriale*

PARAMETRI

Richiesta		
Codice Funzione	1 Byte	0x08
Codice Sottofunzione	2 Bytes	
Dati	N° x 2 Bytes	
Risposta		
Codice Funzione	1 Byte	0x08
Codice Sottofunzione	2 Bytes	
Dati	N° x 2 Bytes	
Errore		
Codice Funzione	1 Byte	0x88
Codice Eccezione	1 Byte	01/03/04

ESEMPIO: Funzione 08 - Diagnostica (Solo linea Seriale)

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	08	Codice Funzione	08
Codice Sottofunzione HI	00	Codice Sottofunzione HI	00
Codice Sottofunzione LOW	00	Codice Sottofunzione LOW	00
Dati HI	A5	Dati HI	A5
Dati LOW	37	Dati LOW	37

Figura 18: *Codice Funzione 08 Diagnostica della linea seriale*

Codice Sottofunzione		Descrizione
Hex	Decimale	
00	00	Ritorna Query
01	01	Ripartenza opzioni comunicazione
02	02	Ritorna il registro di diagnostica
03	03	Cambia il delimitatore ADSCII
04	04	Forza in stato di ascolto
05..09		Non Usati
0A	10	Azzera contatori e registro diagnostico
0B	11	Restituisce il contatore dei messaggi
0C	12	Restituisce il contatore degli errori di comunicazione
0D	13	Restituisce il numero delle eccezioni Modbus
0E	14	Restituisce il numero di richieste inviate allo Slave
0F	15	Restituisce il numero di richieste a cui lo Slave non ha risposto
10	16	Restituisce il numero di NAK dello Slave
11	17	Restituisce il numero di eccezioni di occupato dello Slave
12	18	Restituisce il numero di messaggi che hanno causato un sovraccarico.
13	19	Riservato
14	20	Azzera il contatore di sovraccarico ed i flags

Figura 18b: *Codice Funzione 08 Sotto opzioni*

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x0B

Risposta

Codice Funzione	1 Byte	0x0B
Stato	2 Bytes	da 0x0000 a 0xFFFF
Contatore Eventi	2 Bytes	da 0x0000 a 0xFFFF

Errore

Codice Funzione	1 Byte	0x8B
Codice Eccezione	1 Byte	01/04

ESEMPIO:Funzione 11 - Acquisisci il contatore di eventi della seriale

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	0B	Codice Funzione	0B
		Stato HI	FF
		Stato LOW	FF
		Cont. Eventi HI	01
		Cont. Eventi LOW	08

Figura 19: *Codice Funzione 11 Acquisisce il contatore eventi della seriale*

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0X0C

Risposta		
Codice Funzione	1 Byte	0X0C
Numero Bytes Restituiti	1 Byte	N°
Stato	2 Bytes	Da 0x0000 a 0xFFFF
Contatore Eventi	2 Bytes	Da 0x0000 a 0xFFFF
Contatore Messaggi	2 Bytes	Da 0x0000 a 0xFFFF
Contatore Eventi	(N-6) x 1 Byte	

Errore

Codice Funzione	1 Byte	0x8C
Codice Eccezione	1 Byte	01/04

ESEMPIO:Funzione 12 - Acquisisci il LOG della Seriale

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	0C	Codice Funzione	0C
		Numero Bytes	08
		Stato HI	00
		Stato LOW	00
		Eventi HI	01
		Eventi LOW	08
		Messaggi HI	01
		Messaggi LOW	21
		Evento 0	20
		Evento 1	00

Figura 20: *Codice Funzione 12 Acquisisce il LOG eventi della linea seriale*

questo stato lo Slave non risponde ai comandi che gli vengono inviati escluso 08 01 per toglierlo da tale stato.
Tutto il traffico a lui diretto viene monitorato. Il suo watchdog Timer viene azzerato ed i suoi controlli bloccati.

Sottofunzione 00 04
Dati Richiesta 00 00
Dati Risposta Nessuna risposta

10 Azzera contatori e registro diagnostico

Azzera i contatori ed il registro diagnostico a 16 Bits.

Sottofunzione 00 0A
Dati Richiesta 00 00
Dati Risposta Eco della richiesta

11 Restituisce il contatore dei messaggi

Restituisce quanti messaggi il device remoto ha individuato da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione 00 0B
Dati Richiesta 00 00
Dati Risposta Numero Totale dei Messaggi ricevuti

12 Restituisce il contatore degli errori di comunicazione.

Restituisce il numero di errori CRC individuati da:

- L'ultima ripartenza/reset,
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione 00 0C
Dati Richiesta 00 00
Dati Risposta Numero Totale degli errori CRC

13 Restituisce il numero delle eccezioni ModBus

Restituisce il numero delle eccezioni ModBus inviate come risposta dallo Slave da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione 00 0D
Dati Richiesta 00 00
Dati Risposta Numero Totale Eccezioni

14 Restituisce il numero di richieste/messaggi inviati allo Slave

Restituisce il numero di messaggi inviati allo Slave in oggetto del test o di tipo Broadcast da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione	00 0E
Dati Richiesta	00 00
Dati Risposta	Numero Totale dei Messaggi inviati

15 Restituisce il numero di messaggi a cui lo Slave non ha risposto

Restituisce il numero di messaggi inviati dal Master a cui lo Slave non ha risposto. Non sono comprese le eccezioni che vengono considerate come risposte. Questo da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione	00 0F
Dati Richiesta	00 00
Dati Risposta	N° Totale dei Messaggi senza risposta

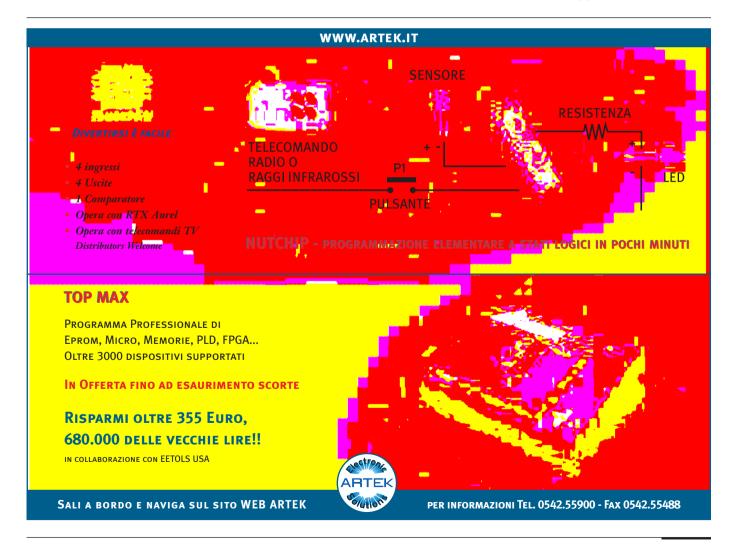
16 Restituisce il numero di NAK dello Slave

Restituisce il numero di eccezioni NAK (Negative Acknowledge) inviati dallo Slave come risposta alle Richieste/Messaggi inviati dal Master. Questo da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione	00 10
Dati Richiesta	00 00
Dati Risposta	Numero Totale dei NAK

17 Restituisce il numero di eccezioni di occupato dello Slave.

Restituisce il numero di messaggi a cui lo Slave ha

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x0F
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Coils	2 Bytes	Da 0x0001 a 0x07B0
Numero di Bytes	1 Byte	N°
Valori	N° x 1 Bytes	

Risposta

Codice Funzione	1 Byte	0x0F
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Coils	2 Bytes	Da 0x0001 a 0x07B0

Errore

Codice Funzione	1 Byte	0x8F
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 15 - Scrivi multipli Coils

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	0F	Codice Funzione	0F
Indirizzo HI	00	Numero Bytes	00
Indirizzo LOW	13	Valore Indirizzo 27-20	13
Numero di Coils HI	00	Valore Indirizzo 35-28	00
Numero di Coils LOW	0A	Valore Indirizzo 38-36	0A
Numero Bytes	02		
Valori Coils HI	CD		
Valori Coils LOW	01		

Figura 21: *Codice Funzione 15 Scrivi Coils multipli*

risposto con occupato (Busy) da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione 00 11
Dati Richiesta 00 00
Dati Risposta Numero Totale Busy

18 Restituisce il numero di messaggi che hanno causato un sovraccarico.

Restituisce il numero dei messaggi che hanno causato un sovraccarico nell'invio dei caratteri costituenti il messaggio. Questo è causato dal fatto che la velocità a cui vengono inviati i singoli caratteri costituenti il messaggio è superiore alla capacità dello Slave di memorizzarli. Può significare una errata configurazione della Seriale o un malfunzionamento della stessa. Questa da:

- L'ultima ripartenza/reset.
- Ultimo Reset dei contatori.
- Ultimo Avviamento.

Sottofunzione 00 12
Dati Richiesta 00 00
Dati Risposta Numero Totale dei Messaggi

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x10
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri	2 Bytes	Da 0x0001 a 0x0078
Numero di Bytes	1 Byte	2 x N°
Valori Registri	N° x 2 Bytes	Valore

Risposta

Codice Funzione	1 Byte	0x10
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri	2 Bytes	Da 0x0001 a 0x0078

Errore

Codice Funzione	1 Byte	0x90
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 16 - Scrivi Registri Multipli

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	10	Codice Funzione	10
Indirizzo HI	00	Indirizzo HI	00
Indirizzo LOW	01	Indirizzo LOW	01
Numero di Registri HI	00	Numero di Registri HI	00
Numero di Registri LOW	02	Numero Di Registri LOW	02
Numero di Bytes	04		
Valore Registro HI	00		
Valore Registro LOW	0A		
Valore Registro HI	01		
Valore Registro LOW	02		

Figura 22: *Codice Funzione 16 Scrivi Registri multipli*

20 Azzera il contatore di sovraccarico ed i flag.

Serve per azzerare il contatore di sovraccarico caratteri ed i flag di errore.

Sottofunzione 00 14
Dati Richiesta 00 00
Dati Risposta Eco della richiesta

Codice Funzione 11 (0x0B) Acquisisci il contatore eventi della Seriale.

Questa funzione ha lo scopo di leggere lo stato ed il relativo contatore degli eventi andati a buon fine della seriale. Il contatore viene incrementato per ogni messaggio andato a buon fine. Non viene incrementato per le risposte che contengono delle eccezioni. Il contatore viene resettato mediante la funzione 08 (00-01) e 08 (00-0A). Lo stato viene posto a FF FF nel caso che lo Slave sia in stato di Occupato (Busy) facendo capire che sta elaborando e quindi non è disponibile, altrimenti contiene 00 00 indicante che è pronto a ricevere altre richieste.

Codice Funzione 12 (0x0C) Acquisisci il LOG eventi della Seriale.

Questa funzione legge lo Stato, il contatore di Eventi, di Messaggi. Lo Stato ed il contatore degli Eventi viene anche restituito dalla precedente funzione (11). Si tratta di una funzione che permette di acquisire molti stati senza dover utilizzare le funzioni che vi accedono singolarmente. La dizione LOG fa comprendere che restituisce uno situazione complessiva dei vari stati dello Slave nell'ambito della comunicazione Seriale. I Bytes che possono sono nell'ordine, 2 per lo Stato, 2 per il contatore

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x11

Risposta

Codice Funzione	1 Byte	0x11
Numero Bytes Restituiti	1 Byte	
ID Slave	Dipende dal Device	
Indicatore di funzionamento	1 Byte	0x00 = OFF , 0xFF = ON
Dati addizionali		

Errore

Codice Funzione	1 Byte	0x91
Codice Eccezione	1 Byte	01/04

ESEMPIO: Funzione 17 - Caratteristiche Slave ID (Solo linea seriale)

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	11	Codice Funzione	11
		Numero Bytes	Dipende dal Device
		ID Slave	Dipende dal Device
		Indic. Funzionamento	0x00 o 0xFF
		Dati Addizionali	Dipende dal Device

Figura 23: Codice Funzione 17 Legge le caratteristiche dello Slave Via sua ID

degli eventi, 2 per il contatore dei messaggi e da 0 a 64 per gli eventi. I valori che vengono restituiti, sono stati precedentemente salvati in ordine cronologico con in cima quello più recente ed in fondo quello più vecchio sino ad un massimo di 65. Ogni nuovo evento fa scorrere verso il basso di una posizione.
Vediamo ora i vari tracciati di Bits, come si identificano ed il significato dei vari Bits di ogni tracciato.

- **Evento di Ricezione**
  Viene generato quando la richiesta da parte del Master viene ricevuta e prima che lo stesso venga elaborato da parte dello Slave. Il Bit 7 posto ad 1 lo identifica. Gli altri Bit sono posti nei rispettivi stati a fronte delle condizioni vere o false degli stessi.

Bit	Contenuto
0	Non Utilizzato
1	Errore di Comunicazione
2	Non Utilizzato
3	Non Utilizzato
4	Sovraccarico Caratteri
5	Attualmente in stato di Ascolto
6	Ricezione di tipo Broadcast
7	1 - Serve ad indicare che il tracciato è di Ricezione

- **Evento di Trasmissione**
  Viene generato quando lo Slave ha elaborato la richiesta ricevuta dal master. Questo avviene sia nel caso di una corretta elaborazione che, in caso di eccezione o di non risposta al Master. Viene identificato dal Bit 6 posto ad 1 e dal Bit 7 posto a 0. Gli altri Bit sono posti nei rispettivi stati a fronte delle condizioni vere o false degli stessi.

Bit	Contenuto
0	Lettura eccezione spedita (ecc. 1-3)
1	Lo Slave interrompe la spedizione dell'eccezione (ecc. 4)
2	Lo Slave è occupato nella spedizione di una eccezione (ecc. 5-6)
3	Lo Slave sta spedendo un'eccezione di tipo NAK (ecc. 7)
4	Time Out di scrittura
5	Attualmente in stato di Ascolto
6	1
7	0

- **Stato di Ascolto**
  Viene generato quando lo Slave entra nello stato di Ascolto. Il valore contenuto è 04 Hex.
- **Device inizializzato, Ripartenza**
  Viene generato quando la porta di comunicazione seriale viene fatta ripartire. Lo Slave viene fatto ripartire/riavviare mediante la funzione 08 (00 01). Essa pone lo Slave in uno stato di

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x14
Numero di Bytes	2 Bytes	Da 0x07 a 0xF5
Sottorichiesta X, Tipo	2 Bytes	06
Sottorichiesta X, Numero File	2 Bytes	Da 0x0000 a 0xFFFF
Sottorichiesta X, Numero Record	2 Bytes	Da 0x0000 a 0x270F
Sottorichiesta X, Lunghezza Record	2 Bytes	N°
Sottorichiesta X+1, ...		

Risposta

Codice Funzione	1 Byte	0x14
Numero Bytes Restituiti	1 Byte	Da 0x07 a 0xF5
Sottorichiesta X, Lunghezza File	1 Byte	Da 0x07 a 0xF5
Sottorichiesta X, Tipo	1 Byte	06
Sottorichiesta X, Dati Record	N° x 2 Bytes	
Sottorichiesta X+1, ...		

Errore

Codice Funzione	1 Byte	0x94
Codice Eccezione	1 Byte	01/02/03/04/08

ESEMPIO: Funzione 20 / 6 - Leggi Record File

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	14	Codice Funzione	14
Numero di Bytes	0C	Numero Bytes Restituiti	0E
Sottorichiesta 1, Tipo	06	Sottorichiesta 1, Lunghezza File	05
Sottorichiesta 1, Numero File HI	00	Sottorichiesta 1, Tipo	06
Sottorichiesta 1, Numero File LOW	04	Sottorichiesta 1, Dati Record HI	0D
Sottorichiesta 1, Numero Record HI	00	Sottorichiesta 1, Dati Record LOW	FE
Sottorichiesta 1, Numero Record LOW	01	Sottorichiesta 1, Dati Record HI	00
Sottorichiesta 1, Lunghezza Record HI	00	Sottorichiesta 1, Dati Record LOW	20
Sottorichiesta 1, Lunghezza Record LOW	02	Sottorichiesta 2, Lunghezza File	05
Sottorichiesta 2, Tipo	06	Sottorichiesta 2, Tipo	06
Sottorichiesta 2, Numero File HI	00	Sottorichiesta 2, Dati Record HI	33
Sottorichiesta 2, Numero File LOW	03	Sottorichiesta 2, Dati Record LOW	CD
Sottorichiesta 2, Numero Record HI	00	Sottorichiesta 2, Dati Record HI	00
Sottorichiesta 2, Numero Record LOW	09	Sottorichiesta 2, Dati Record LOW	40
Sottorichiesta 2, Lunghezza Record HI	00		
Sottorichiesta 2, Lunghezza Record LOW	02		

Figura 24: Codice Funzione 20 Legge Record File

Continua per Errore o Fermati per Errore ed il resto del LOG viene azzerato. Il valore contenuto è 00 Hex.

Nell'esempio riportato in Figura 20 possiamo osservare che lo stato contiene 00 00 indicante che lo Slave è pronto per altre elaborazioni. Il contatore degli eventi contiene 01 08 cioè 264 in decimale. Il contatore dei messaggi contiene 01 21 che equivale a 289 in decimale. Seguono i vari eventi. In posizione 0 troviamo 20 che indica che il sistema era entrato in stato di ascolto. Mentre in posizione 1, antecedente alla 0 troviamo 00 che, indica che era stato attivato il riavvio.

Codice Funzione 15 (0x0F) Scrivi Coils multipli
Utilizzata per porre a ON o OFF i Coils di una sequenza di Coils. L'indirizzo iniziale parte da 0 come anche il Coil da modificare. Lo stato ON equivale a porre il corrispondente Bit a 1, mentre lo stato OFF a porre il Bit a 0. La risposta dello Slave comprende il codice funzione. L'indirizzo di partenza e la quantità di Coils da modificare.
Nell'esempio riportato in Figura 21 vengono modificati 10 coils a partire dal coil 20 in poi. La richiesta da parte del Master pone il Byte HI (Coils 27-20 valore CD Hex) per primo ed il Byte LOW (Coils 29-28 valore 01 Hex) per secondo. I rispettivi Bits riferiti ai Coils da modificare sono cosi disposti:

```
1  1  0  0  1  1  0  1  0  0  0  0 0  0  0  1
27 26 25 24 23 22 21 20 -  -  -  -  -  -  29 28
```

I Bits non utilizzati sono posti a 0.

Codice Funzione 16 (0x10) Scrivi Registri Multipli
Utilizzata per scrivere un blocco contiguo di registri (da 1 a circa 120). Nel richiesta viene specificato l'indirizzo di partenza, il numero dei registri , il numero dei Bytes che compongono i dati ed infine i dati veri e propri.
La risposta da parte dello Slave restituisce oltre il codice funzione, l'indirizzo di partenza e la quantità dei registri modificati. Ogni registro occupa 2 Bytes essendo a 16 Bits.

Codice Funzione 17 (0x11) Legge le caratteristiche dello Slave Via sua ID
Legge la descrizione dello Slave, il tipo, lo stato corrente ed altre informazioni.

Codice Funzione 20 - 6 (0x14 – 0x06) Leggi Record File
Utilizzata per gestire la lettura sottoforma di file e records. Il file è organizzato in records e può contenerne un massimo di 10000, indirizzabili da 0000 a 9999 (0x0000 a 0x270F in esadecimale).
La richiesta di lettura prevede gruppi multipli ed ogni gruppo può anche essere non contiguo al precedente.
Ogni gruppo è definito da:

- Tipo Riferimento: 1 Byte con valore 0x06
- Numero File: 2 Bytes
- Record da cui iniziare: 2 Bytes
- Numero Records: 2 Bytes

La quantità di dati combinati tra di loro non può superare i 256 Bytes previsti come lunghezza massima per una richiesta/Messaggio dal ModBus.
La normale risposta prevede una serie di sotto-risposte per ogni serie di sotto-richieste fatte dal Master.

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x15
Numero di Bytes	1 Byte	Da 0x07 a 0xF5
Sottorichiesta X, Tipo	1 Byte	06
Sottorichiesta X, Numero File	2 Bytes	Da 0x0000 a 0xFFFF
Sottorichiesta X, Numero Record	2 Bytes	Da 0x0000 a 0x270F
Sottorichiesta X, Lunghezza Record	2 Bytes	N°
Sottorichiesta X, Dati Record	N° x 2 Bytes	
Sottorichiesta X+1, ...		

Risposta

Codice Funzione	1 Byte	0x15
Numero Bytes Restituiti	1 Byte	
Sottorichiesta X, Tipo	1 Byte	06
Sottorichiesta X, Numero File	2 Bytes	Da 0x0000 a 0xFFFF
Sottorichiesta X, Numero Record	2 Bytes	Da 0x0000 a 0xFFFF
Sottorichiesta X, Lunghezza Record	2 Bytes	Da 0x0000 a 0xFFFF N°
Sottorichiesta X, Dati Record	N° x 2 Bytes	
Sottorichiesta X+1, ...		

Errore

Codice Funzione	1 Byte	0x95
Codice Eccezione	1 Byte	01/02/03/04/08

ESEMPIO: Funzione 21 / 6 - Scrivi Record File

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	15	Codice Funzione	15
Numero di Bytes	0D	Numero di Bytes	0D
Sottorichiesta 1, Tipo	06	Sottorichiesta 1, Tipo	06
Sottorichiesta 1, Numero File HI	00	Sottorichiesta 1, Numero File HI	00
Sottorichiesta 1, Numero File LOW	04	Sottorichiesta 1, Numero File LOW	04
Sottorichiesta 1, Numero Record HI	00	Sottorichiesta 1, Numero Record HI	00
Sottorichiesta 1, Numero Record LOW	07	Sottorichiesta 1, Numero Record LOW	07
Sottorichiesta 1, Lunghezza Record HI	00	Sottorichiesta 1, Lunghezza Record HI	00
Sottorichiesta 1, Lunghezza Record LOW	03	Sottorichiesta 1, Lunghezza Record LOW	03
Sottorichiesta 1, Dati Record HI	06	Sottorichiesta 1, Dati Record HI	06
Sottorichiesta 1, Dati Record LOW	AF	Sottorichiesta 1, Dati Record LOW	AF
Sottorichiesta 1, Dati Record HI	04	Sottorichiesta 1, Dati Record HI	04
Sottorichiesta 1, Dati Record LOW	BE	Sottorichiesta 1, Dati Record LOW	BE
Sottorichiesta 1, Dati Record HI	10	Sottorichiesta 1, Dati Record HI	10
Sottorichiesta 1, Dati Record LOW	OD	Sottorichiesta 1, Dati Record LOW	OD

Figura 25: Codice Funzione 21 Scrive Record File

Codice Funzione 21 - 6 (0x15 – 0x06) Scrivi Record File
Utilizzata per gestire la scrittura sottoforma di file e records. Il file è organizzato in records e può contenerne un massimo di 10000, indirizzabili da 0000 a 9999 (0x0000 a 0x270F in esadecimale). La richiesta di lettura prevede gruppi multipli ed ogni gruppo può anche essere non contiguo al precedente.
Ogni gruppo è definito da:

- Tipo Riferimento: 1 Byte con valore 0x06
- Numero File: 2 Bytes
- Record da cui iniziare: 2 Bytes
- Numero Records: 2 Bytes
- I dati che vengono scritti: 2 Bytes

La quantità di dati combinati tra di loro non può superare i 256 Bytes previsti come lunghezza massima per una richiesta/Messaggio dal ModBus.
La normale risposta è un eco della richiesta fatta dal Master.

Codice Funzione 22 (0x16) Maschera Registro di Scrittura
Viene utilizzata per modificare il contenuto di uno specifico registro, utilizzando una combinazione di maschere AND ed OR. Essa può essere utilizzata per settare o resettare I Bits del registro indirizzato. Lavora con gli Holding register. L'algoritmo su cui si basa è:

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x16
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Maschera AND	2 Bytes	Da 0x0000 a 0xFFFF
Maschera OR	2 Bytes	Da 0x0000 a 0xFFFF

Risposta

Codice Funzione	1 Byte	0x16
Indirizzo Iniziale	2 Bytes	Da 0x0000 a 0xFFFF
Maschera AND	2 Bytes	Da 0x0000 a 0xFFFF
Maschera OR	2 Bytes	Da 0x0000 a 0xFFFF

Errore

Codice Funzione	1 Byte	0x96
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO:Funzione 22 - Maschera Registri Scrittua

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	16	Codice Funzione	16
Indirizzo HI	00	Indirizzo HI	00
Indirizzo LOW	04	Indirizzo LOW	04
Maschera AND HI	00	Maschera AND HI	00
Maschera AND LOW	f2	Maschera AND LOW	f2
Maschera OR HI	00	Maschera OR HI	00
Maschera OR LOW	25	Maschera OR LOW	25

Figura 26: *Codice Funzione 22 Maschera registro di scrittura*

Risultato = (Contenuto corrente registro AND Maschera_AND) OR (Maschera_OR AND Maschera_AND_NEGATO)

Esempio:
Contenuto corrente registro = 12 = 0001 0010
Maschera AND = F2 = 1111 0010
Maschera OR = 25 = 0010 0101
Maschera AND NEGATA = 0D = 0000 1101
Risultato = 17 = 0001 0111

Nota: se la maschera OR vale zero, il risultato è semplicemente un AND logico con il contenuto corrente ed una maschera AND. Se la maschera AND vale zero il risultato è uguale alla maschera OR.

Codice Funzione 23 (0x17) Leggi/Scrivi Registri Multipli
Combina una operazione di lettura ed una operazione di scrittura in un unica transazione ModBus. La scrittura viene fatta prima della lettura. La richiesta specifica l'indirizzo di partenza della lettura, il numero dei registri da leggere, l'indirizzo iniziale della scrittura, il numero di registri in cui scrivere, la quantità in Bytes interessati alla scrittura ed infine i valori da scrivere.

Codice Funzione 24 (0x18) Leggi Coda FIFO
Serve per leggere il contenuto della coda FIFO (First-In-First-Out in Italiano Il primo che entra è il primo che esce) dei registri dello Slave. Essa ritorna il numero dei registri presenti nella coda, seguito dalla coda vera e propria contenente i valori dei registri. Sino ad un massimo di 32 registri possono essere presenti nella coda. Se il numero supera 31 viene generata una eccezione 03. La lettura della coda FIFO non ne azzera il contenuto.

Codice Eccezione risposte
In Figura 29 troviamo il significato dei codici di eccezione restituiti dallo Slave in caso di errore.

CONCLUSIONE

Questo e quanto in riferimento al protocollo ModBus. Abbiamo fatto una rapida carrellata di che cosa sia, delle sue caratteristiche e dei codici funzioni utilizzati nel suo ambito. Questo articolo non ha avuto la pretesa di essere stato esaustivo

riguardo l'argomento anche perchè non abbiamo parlato del ModBus TCP/IP. Esso ha voluto semplicemente introdurre cosa sia il protocollo ModBus a quanti vi si avvicinino per la prima volta, e dato che la documentazione che si trova sotto Internet è prevalentemente in Inglese trovare tale spiegazione in Italiano ne semplifica la comprensione.

Bene per questa puntata e tutto, arrivederci alla prossima dove implementeremo la versione ASCII in Visual Basic.

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x17
Indirizzo Iniziale Lettura	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri da Leggere	2 Bytes	Da 0x0001 a appross. 0x0076
Indirizzo Iniziale Scrittura	2 Bytes	Da 0x0000 a 0xFFFF
Numero di Registri da Scrivere	2 Bytes	Da 0x0001 a appross. 0x0076
Numero Bytes da Scrivere	1 Byte	2 x N°
Valori scrittura Registri	N° x 2 Bytes	

Risposta

Codice Funzione	1 Byte	0x17
Numero Bytes Restituiti	1 Byte	2 x N°
Stato Registri Richiesti	N° x 2 Bytes	n = N o N + 1

Errore

Codice Funzione	1 Byte	0x97
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 23 - Leggi/Scrivi Registri Multipli

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	17	Codice Funzione	17
Indirizzo Iniziale Lettura HI	00	Numero Bytes	0C
Indirizzo Iniziale Lettura LOW	03	Stato Registri Richiesti HI	00
Numero di Registri da Leggere HI	00	Stato Registri Richiesti LOW	FE
Numero di Registri da Leggere LOW	06	Stato Registri Richiesti HI	0A
Indirizzo Iniziale Scrittura HI	00	Stato Registri Richiesti LOW	CD
Indirizzo Iniziale Scrittura LOW	0E	Stato Registri Richiesti HI	00
Numero di Registri da Scrivere HI	00	Stato Registri Richiesti LOW	01
Numero di Registri da Scrivere LOW	03	Stato Registri Richiesti HI	00
Numero Bytes da Scrivere	06	Stato Registri Richiesti LOW	03
Valori scrittura Registri HI	00	Stato Registri Richiesti HI	00
Valori scrittura Registri LOW	FF	Stato Registri Richiesti LOW	0D
Valori scrittura Registri HI	00	Stato Registri Richiesti HI	00
Valori scrittura Registri LOW	FF	Stato Registri Richiesti LOW	FF
Valori scrittura Registri HI	00		
Valori scrittura Registri LOW	FF		

Figura 27: *Codice Funzione 23 Legge/Scrive Registri Multipli*

PARAMETRI

Richiesta

Codice Funzione	1 Byte	0x18
Indirizzo Iniziale FIFO	2 Bytes	Da 0x0000 a 0xFFFF

Risposta

Codice Funzione	1 Byte	0x18
Numero Bytes Restituiti	2 Bytes	N°
Contatore FIFO	2 Bytes	< = 31
Valore Registro FIFO	N° x 2 Bytes	

Errore

Codice Funzione	1 Byte	0x98
Codice Eccezione	1 Byte	01/02/03/04

ESEMPIO: Funzione 24 - Leggi Coda FIFO

Richiesta		Risposta	
Nome campo	Valore HEX	Nome Campo	Valore HEX
Codice Funzione	18	Codice Funzione	18
Indirizzo Puntatore FIFO HI	04	Numero Bytes HI	00
Indirizzo Puntatore FIFO LOW	DE	Numero Bytes LOW	06
		Contatore FIFO HI	00
		Contatore FIFO LOW	02
		Valore Registro FIFO HI	01
		Valore Registro FIFO LOW	B8
		Valore Registro FIFO HI	12
		Valore Registro FIFO LOW	84

Figura 28: *Codice Funzione 24 Legge Coda FIFO*

Codici di Eccezione MODBUS		
Codice	**Nome**	**Descrizione**
01	**Funzione illegale**	**Il codice funzione ricevuto dallo Slave non è applicabile. Questo può essere causato dal fatto che, la stessa non è implementata. Oppure che lo Slave si trova in uno stato che ne permette la gestione.**
02	**Indirizzo illegale**	**L'indirizzo ricevuto dallo Slave non rientra tra quelli previsti dallo stesso. Ad esempio in uno Slave dotato di 100 registri una richiesta di lettura a partire dal registro numero 96 per 5 registri, supera le dimensioni.**
03	**Valori illegali**	**Valori non appartenti al range previsto per lo Slave in riferimento al registro interessato dall'operazione.**
04	**Errore device Slave**	**E' avvenuto un errore irricoverabile, per cui è impossibile portare a terminae la richiesta.**
05	**Acknowledge**	**Lo Slave sta elaborando la richiesta ma, la stessa richiede del tempo per poter essere completata. Lo Slave invia quindi questo tipo di eccezione per per prevenire un errore di Time Out. Il Master controlla nel prossimo ciclo che lo Slave abbia finito.**
06	**Device Slave Occupato**	**Lo Slave è impegnato in una elaborazione molto lunga. L'invio di questa eccezione comunica la Master di non inviare altre richiesto sino a quando lo Slave torna disponibile.**
08	**Errore memoria (Parità)**	**Errore specializzato che indica che un blocco di dati (Record) non ha passato il controllo di integrità degli stessi.**

Figura 29: *Codici Eccezioni*

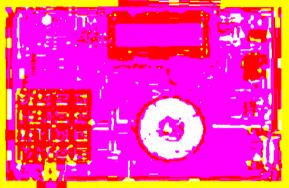
GENERATORE DI FUNZIONI PROGRAMMABILE

di Agostino Rolando
a.rolando@farelettronica.com

Il circuito qui presentato costituisce un utile strumento da laboratorio compatto e versatile, in grado di generare segnali di qualsiasi forma. Ma è anche un'ottima occasione per sperimentare con le CPLD…

Il progetto consiste in un generatore di funzioni le cui caratteristiche sono configurabili per mezzo dell'interfaccia JTAG presentata sul numero di febbraio 2004, da collegare al PC tramite la porta parallela.
Il circuito e' in grado di generare un set predefinito di forme d'onda, che vengono selezionate per mezzo di una tastiera e di visualizzare messaggi su un display LCD.
Contiene inoltre un'interfaccia seriale allo scopo di generare dei pattern di test per linee seriali e un driver di potenza in grado di pilotare carichi induttivi, come solenoidi o motori elettrici.
In allegato all'articolo, a disposizione sul sito di Fare Elettronica, viene riportato il codice sorgente della procedura Verilog di controllo che gestisce il display, la tastiera e consente di generare alcune forme d'onda.

DESCRIZIONE DEL CIRCUITO

In figura 1 è raffigurato lo schema elettrico del generatore di funzioni.

Chip programmabile
Il circuito è incentrato sull' integrato IC03, CPLD *Xilinx* in package PLCC a 44 pin, alimentata a 3.3 V, la quale è in grado di accettare anche ingressi a livello TTL.
Il dispositivo viene resettato all'accensione per mezzo della rete R-C composta da R07 e C10.
Il clock di funzionamento è fornito dall'oscillatore IC04. Per questa applicazione ho scelto una frequenza di 1.843.200 Mhz, ma il dispositivo accetta anche valori più elevati, fino ad un massimo di 178 Mhz.
La programmazione del chip viene fornita attraverso il connettore JTAG J04.
Ai pin IO11 e IO12 sono collegati due led, per monitorare l'attività del sistema.
All'ingresso IO14 della CPLD è collegato il transistor TR01, in funzione di buffer per un eventuale ingresso di trigger esterno, alla morsettiera M01. Su questo ingresso è posta la predisposizione P01 che permette di selezionare un accoppiamento in AC o in DC.

Tastiera 4x4
La CPLD ha otto pin dedicati alla gestione di una tastiera a matrice X-Y da 16 tasti, che va inserita sui connettori a striscia J06 e J07. In figura 1A sono visibili le dimensioni e il pin-out della tastiera.

Display lcd
L'integrato IC05 è un buffer 74AC245 e serve a interfacciare la CPLD con il display LCD (da inserire al connettore J05) e con il convertitore D/A IC07, entrambi alimentati a 5 V.
Il display è del tipo standard a 2 righe da 16 caratteri. Viene gestito dalla CPLD con un'opportuna procedura Verilog.

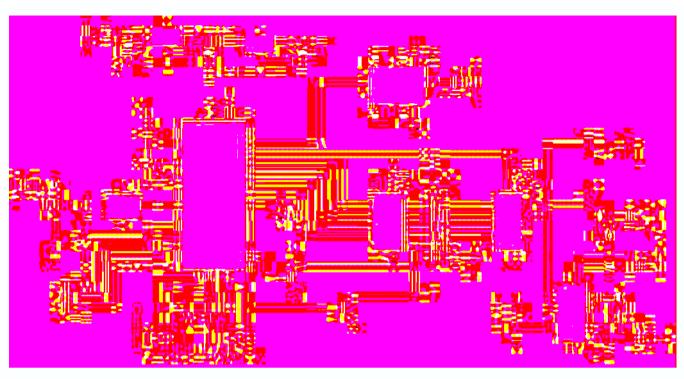



Figura 1: *Schema elettrico*

Alimentazione

L'alimentazione al sistema è doppia: al connettore J01 si immette la tensione che alimenta la circuiteria digitale, che è poi la gran parte del circuito. La tensione di ingresso può variare da 7 a 12 V. Le tensioni a 5 V e a 3.3 V sono ricavate per mezzo dei rispettivi regolatori serie IC01 e IC02.
Per quanto riguarda invece la sezione di pilotaggio di potenza, che fa capo all' integrato driver IC08, questa richiede un'alimentazione aggiuntiva, fino a 46 V, che viene fornita per mezzo del connettore J03.

Figura 1a: *Dimensioni e pin-out della tastiera*

Integrato driver

Il dispositivo L298 (figura 2) è un circuito integrato monolitico in contenitore Multiwatt a 15 pin. È un bridge driver da alta tensione e alta corrente, comandabile con livelli logici TTL ed è in grado di pilotare carichi induttivi come

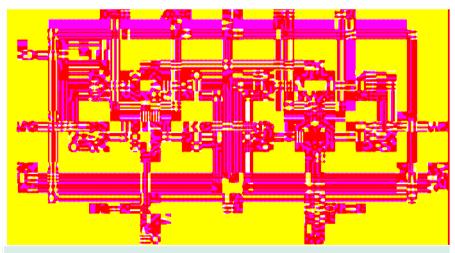



Figura 2: *Schema a blocchi dell'integrato L 298*

Figura 3: *Schema a blocchi e curva di trasferimento dell'integrato CA3338*

relé, solenoidi, motori in corrente continua o passo-passo.
Possiede due ingressi di abilitazione (ENABLE A, ENABLE B).
Gli emettitori dei transistor lato inferiore di ciascun bridge sono collegati insieme e i corrispondenti terminali esterni possono essere usati per connessione a una resistenza di *sensing* esterna (nello schema elettrico, R33 e R34). La parte logica viene alimentata a 5 V, al pin di alimentazione *Vss*, distinto da quello di alimentazione di potenza (Vs).

Per incrementare la corrente erogabile in uscita, ho collegato insieme i pin di ingresso dei transistor 1 e 4 (IN1, IN4) e le rispettive uscite OUT1, OUT4.
Così facendo, IC08 può pilotare un carico di tipo induttivo, con un assorbimento massimo di 2 A.

Convertitore D/A

L'integrato CA3338 è un convertitore digitale-analogico di tipo flash da 50 megasample al secondo, funzionante a singola alimentazione (5 V).
Per questa applicazione viene utilizzata la versione in package SOIC a 16 pin.
Lo schema a blocchi e la caratteristica di trasferimento del convertitore sono riportati in figura 3.

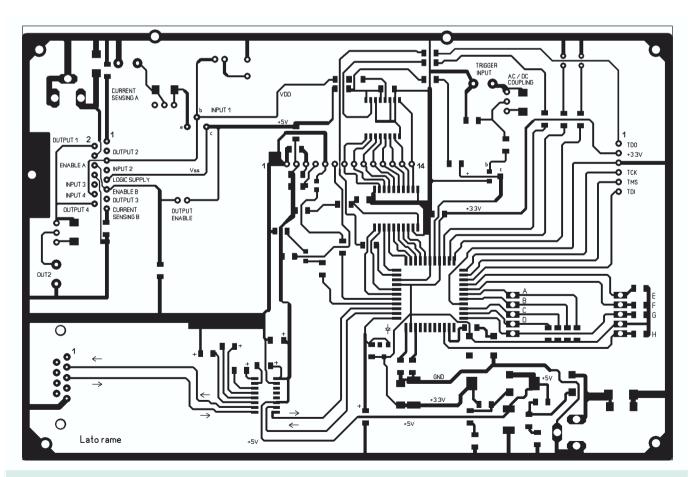



Figura 4: *Circuito stampato in scala 1:1 (lato rame)*

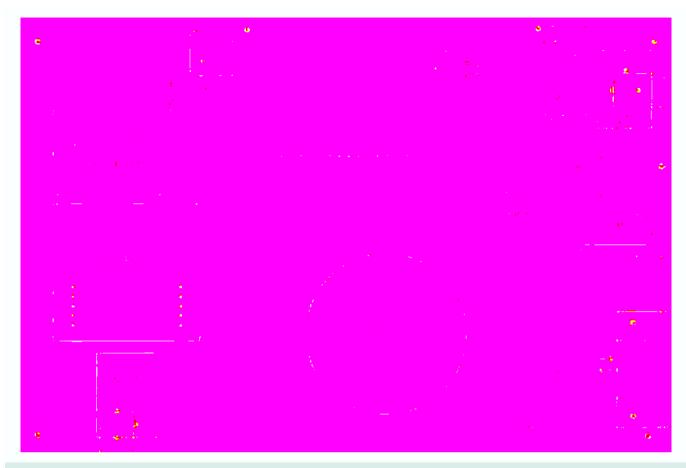



Figura 5: *Schema di montaggio lato componenti*

Uscita

L'uscita analogica, proveniente dal convertitore D/A viene bufferata con il transistor TR04, in configurazione a emitter-follower, ed è riportata alla morsettiera M02.

Per quanto riguarda l'uscita di potenza, si può prelevare alla morsettiera M03.

Per entrambe, come per l'ingresso di trigger, ho previsto delle

predisposizioni a ponticello (rispettivamente P02 e P03) per selezionare l'accoppiamento in AC o in DC.
Il circuito contiene anche un buzzer piezoelettrico (BZ01), attivabile con la predisposizione P04, per poter ascoltare il segnale generato, se nel range di frequenza udibile.

Interfaccia seriale
Lo schema contiene un'interfaccia seriale, realizzata con l'integrato IC06, che ha lo scopo di generare dei pattern per il test di

Elenco componenti			
Sigla	**Valore**	**Sigla**	**Valore**
R01	390 Ω (package 0805)	TR4	Transistor npn 2N1711
R04, R16, R17	330 Ω (package 0805)	IC1, IC2	Regolatore di tensione LM1117H (package TO-252)
R02	15 Ω (package 0805)	IC3	CPLD XC9572XL-10PC44I (package 44 pin PLCC)
R03, R06, R31	270 Ω (package 0805)	IC4	Oscillatore SUNNY Frequenza 1.843200 MHz (SMD)
R05	470 Ω (package 0805)	IC5	Tri-state buffer 74AC245 (package SO20)
R07÷R09, R25, R29, R21	4K7 Ω (package 0805)	IC6	Transceiver MAX232 (package SO16)
R35÷R42	4K7 Ω (package 0805)	IC7	Convertitore D/A Intersil CA3338 (package SO16)
R18	100 Ω (package 0805)	IC8	Driver ST L298HN (package Multiwatt orizz.15 pin)
R20, R26, R28, R30, R33, R34	1 KΩ (package 0805)	J1,J3	Connettore da alimentazione "plug" M per stampato
R22, R32	10K Ω (package 0805)	J2	Connettore a vaschetta, 90 gradi, da stampato , M, 9 pin
RBY (12 pezzi)	0 Ω (package 0805)	J4	Connettore lineare a striscia, 90 gradi, M, 6 pin
C01, C14, C05, C08, C09, C18	0,1 μF 50 V ceramico	J5	Connettore lineare a striscia, F, 15 pin
C02, C13, C15, C17	39 μF 20 V tantalio	J6, J7	Connettore lineare a striscia, F, 5 pin
C03, C04, C16	2,2 μF 20 V tantalio	P1÷P3	Connettore lineare a striscia, M, 3 pin
C06, C07, C10÷C12	1 μF 20 V tantalio	P4, P5	Connettore lineare a striscia, M, 2 pin
LED1	Diodo led verde , 3 mm	M1÷M3	Morsettiera 2 poli
LED2	Diodo led rosso , 3 mm	BZ1	Buzzer piezoel.
D01	Diodo BAV99 (package SOT-23)	KB1	Tastiera 4x4 Grayhill serie 83
TR1÷TR3	Transistor smd npn 2N2222 (package SOT-23)		

Figura 6: Schema di montaggio lato rame

linee di comunicazione. Può essere utilizzata anche per immettere dei comandi da remoto verso il generatore di funzioni.

REALIZZAZIONE PRATICA

Il circuito stampato, visibile in figura 4, è stato realizzato con il metodo a trasferimento di toner, per mezzo di lucidi trasparenti per fotocopiatrici e di un comune ferro da stiro.

I simboli nello schema elettrico denominati RBY identificano resistenze di bypass, che sono state impiegate al fine di realizzare uno stampato monofaccia.

Come contenitore ho utilizzato un porta-cassette video. Lo stampato è sostenuto da quattro distanziatori, saldati al contenitore per mezzo di colla calda.

Il risultato è visibile in figura 11.

FUNZIONALITÀ

Per la gestione del circuito, ho scelto di realizzare una procedura in linguaggio Verilog che potesse stimolare tutte le aree funzionali del circuito.

Figura 7: Packages utilizzati

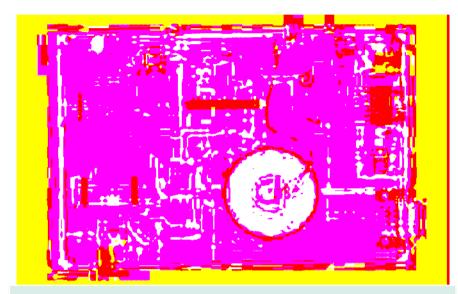



Figura 8: *Prototipo lato componenti senza LCD e tastiera*

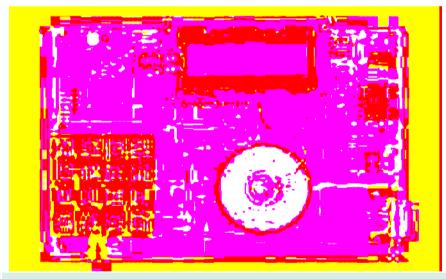



Figura 9: *Prototipo lato componenti*

Generazione di forme d'onda
La procedura scandisce la tastierina e riconosce la pressione dei tasti. In particolare, se viene premuto il tasto "A", viene generata una sinusoide, il tasto "B" invece fa generare un'onda quadra, mentre il tasto "C" fa generare un'onda triangolare. La frequenza del segnale è fissata a 1.8 Khz.

Gestione display
In contemporanea, un altro processo resetta il display LCD e vi scrive la stringa di test "ABCD".

Sincronizzazione con trigger esterno
Per testare l'ingresso di trigger esterno, i due led presenti sul circuito vengono accesi secondo una determinata sequenza di conteggio. Il conteggio viene effettuato in avanti se l'ingresso del trigger esterno è basso, all'indietro se il trigger è alto.

Generazione di pattern seriali
Un'ulteriore applicazione consiste nell'invio di pattern per verificare il corretto funzionamento di dispositivi dotati di linea seriale.
A questo scopo ho realizzato un secondo codice Verilog che realizza la funzione di UART, relativamente alla parte di trasmissione.
La procedura scandisce la tastiera e, in base al tasto premuto, invia un pacchetto di caratteri ASCII (da "0" a "F") sull'uscita seriale, al connettore J02.
Per l'implementazione, ho fatto uso di un clock a 9600 Hz, derivato dal clock di sistema a 1.843200 Hz, che consente di inviare i dati alla velocità di 9600 Baud, con 1 bit di stop e parità dispari.
Per caricare il dato parallelo, che viene inviato poi serialmente, si pone alto il segnale *load_tx*.

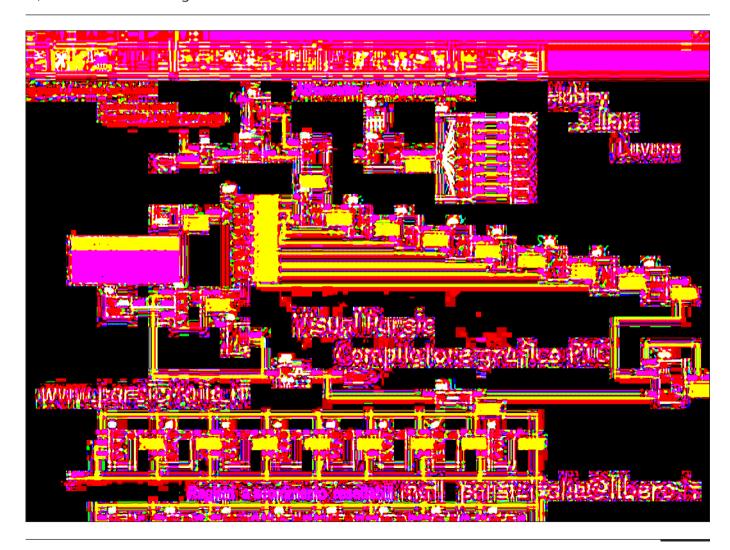
La linea seriale, a riposo, viene tenuta alta. All'inizio della trasmissione, il dato seriale viene posto a livello basso per la durata di un colpo di clock, per significare lo start bit d'inizio della trasmissione, poi, i bit di dato sono inviati uno alla volta, a partire dal meno significativo (D0÷D7). Infine, viene inviato il bit di parità, che viene calcolato come EXOR negato dei bit della parola da inviare. In ultimo, il bit di stop.
Al termine, la linea seriale viene riportata alta, in attesa della prossima trasmissione.

CONCLUSIONI

Il circuito è stato pensato per applicazioni tipiche da laboratorio ed è suscettibile di miglioramenti. Ad esempio, si può pen-

Figura 10: *Prototipo lato rame*

Figura 11: *Prototipo assemblato entro un contenitore per video-cassetta*

sare di sostituire l'oscillatore IC04 con un VCO da comandare con una regolazione manuale, così da avere un segnale di uscita a frequenza variabile. Un'altra possibile applicazione, oltre a quella qui descritta, può consistere nel trasformare il circuito in un frequenzimetro.
I codici sorgenti allegati al progetto sono scaricabili dal sito di Fare Elettronica.

CAMPAGNA ABBONAMENTI 2004-2005

Abbonarsi a Fare Elettronica significa ricevere ogni mese, direttamente a casa, tante idee, consigli e progetti di assoluta novità.

Fare Elettronica trasforma il tuo lavoro o il tuo hobby in una vera passione.

Abbonandoti potrai ricevere la rivista ad un prezzo molto conveniente rispetto a quello di copertina. Pagherai infatti solo € 39,00 invece di € 51,00, con un risparmio di ben € 12,00 e la sicurezza di un prezzo bloccato per un anno.

Inoltre riceverai un **buono sconto del 20%** utilizzabile per il tuo prossimo acquisto su www.farelettronica.com e, con la merce ordinata, riceverai un gradito omaggio di benvenuto!

ABBONARSI A FARE ELETTRONICA CONVIENE.

Abbonarsi subito conviene ancora di più.

Diverse possibilità di abbonamento:

Standard: Il tuo abbonamento personale o aziendale al costo di € 39,00

Regalo: Se sei già abbonato e vuoi regalare un nuovo abbonamento ad un amico, lo pagherai solo € 35,00 (10% di sconto) comunicando il tuo codice

Scuole: Riservato a scuole ed università; ordinando quattro abbonamenti ne riceverai uno in omaggio, pagherai quindi € 156,00 anzichè € 195,00 (20% di sconto)

Come abbonarsi:

Per Posta: scrivere a INWARE Edizioni - Via Cadorna, 27 - 20032 Cormano (MI)
Per Telefono: al numero +39 02.66504794
Per Fax: al numero +39 02.66508225
Via Internet: sul sito www.farelettronica.com alla pagina campagna abbonamenti

(è possibile pagare con bollettino postale, bonifico bancario e carta di credito, maggiori dettagli in ultima pagina)

FARE ELETTRONICA BUNDLES È UNA COMBINAZIONE DI PRODOTTI AD UN PREZZO SCONTATISSIMO.

NON LASCIARTI SCAPPARE IL TUO PREFERITO!

ECCO L'ELENCO AGGIORNATO

Elenco dei prodotti disponibili nei Bundles	Prezzo normale Cad.
❑ Fare Elettronica N. 220/221 Ottobre/Novembre 2003 – Numero Doppio	€ 9,00
❑ Fare Elettronica N. 222 Dicembre 2003	€ 7,50
❑ Fare Elettronica N. 223 Gennaio 2004	€ 7,50
❑ Fare Elettronica N. 224 Febbraio 2004	€ 7,50
❑ Fare Elettronica N. 225 Marzo 2004	€ 7,50
❑ CD-ROM "Annata Fare Elettronica 2003"	€ 25,00
❑ Software Abacom "sPlan Disegno schemi elettrici"	€ 46,80
❑ Software Abacom "sPrint Disegno circuiti stampati"	€ 46,80
❑ Software Abacom "Front Designer Disegno pannelli frontali"	€ 46,80

Elenco dei Bundles disponibili	Prezzo Bunble Cad.
❑ Bundle "A" : 2 Arretrati (qualsiasi) al prezzo speciale di	€ 12,00
❑ Bundle "B" : 3 Arretrati (qualsiasi) al prezzo speciale di	€ 15,00
❑ Bundle "C" : 1 Arretrato (qualsiasi) e il CD-ROM "Annata FE 2003" al prezzo speciale di	€ 30,00
❑ Bundle "D": 1 Arretrato (qualsiasi) e 1 Software Abacom (qualsiasi) al prezzo speciale di	€ 50,00
❑ Bundle "E" : 2 Software Abacom (qualsiasi) al prezzo speciale di	€ 84,00

Indicare il Bundle e i prodotti prescelti mettendo una croce sulle rispettive caselle.

Tutti i Bundle sono comprensivi di IVA e spese di spedizione. Il pagamento è anticipato, e può essere effettuato tramite:

❑ **Bollettino postale** Utilizzare il **C/C N. 22790232** intestato ad **Inware srl**, indicando nella causale **"Abbonamento a Fare Elettronica"**

❑ **Bonifico bancario** Appoggiarlo sulla banca: **Poste Italiane - CIN: Z - ABI: 07601 - CAB: 01600 - C/C: 000022790232**

❑ **Carta di credito** Titolare: ______
Numero: ______ Scadenza: ___ / ___

VISA MasterCard

Cognome ______ Nome ______

Azienda ______

Via ______ CAP ______ Città ______ Prov. ______

Tel. ______ Fax ______ email ______

Privacy. Il trattamento dei dati, in forma automatizzata e con modalità strettamente connesse ai fini, con garanzia di riservatezza, è finalizzato all'invio del presente periodico allo scopo di informare ed aggiornare i lettori e gli operatori del settore elettronico sulle novità che il mercato propone. Potranno essere esercitati i diritti di cui all'articolo 13 della legge 675/96 (accesso, correzione, cancellazione, opposizione al trattamento, ecc.). Il titolare del trattamento dei dati è Inware srl con sede a Cormano (MI) in via Cadorna 27/31. Nel caso si tratti di copia omaggio a titolo promozionale si rende noto che i dati provengono da archivi pubblici. Resta inteso che le informazioni in ns. possesso non saranno in nessun caso cedute a terzi.

GENERATORE PER ARGENTO COLLOIDALE

di Andrea Marani
a.marani@farelettronica.com

Un completo ed efficiente circuito concepito per poter produrre in proprio argento colloidale, le cui peculiarità e caratteristiche vanno dalle ben note doti di antibiotico naturale e disinfettante coadiutore della cicatrizzazione. Queste sono solo un assaggio delle nolteplici capacità dell'argento colloidale, divenuto un best sellers tra i lenimenti naturali..

Molti di voi ricorderanno quando la nonna consigliava alcune pennellate di argento vivo se la gola era arrossata, oppure bere un bel bicchiere di sali di argento per mettere in quiete un intestino un poco nervoso… Ebbene tutto ciò è verissimo al punto che molti sono i casi in cui la medicina utilizza tale prodotto, in particolare per favorire la cicatrizzazione di ferite, garantirne la pulizia e antisetticità, per curare le piaghe tipiche della bocca o infiammazioni delle mucose e via dicendo.
L'argento è di per se un metallo prezioso quindi abbastanza costoso; per poter disporre di argento sottoforma di precipitato o colloide dovremo ricorrere all'elettrolisi utilizzando due bacchette di argento purissimo, non è quindi possibile utilizzare posate o altri oggetti di argento 800 millesimi ma SOLO argento puro 1000/1000 recuperabile presso negozi di chimica o farmacie.
Le bacchette generalmente sono di tipo cilindrico, diametro 3 o 4 millimetri e lunghe circa 5 centimetri (figura 1).
Ad ogni bacchetta si collegherà un polo, positivo o negativo che sia ed il tutto si collegherà ad una fonte di corrente continua. Per provare possono essere ottime tre pile piatte da 4,5V in serie tra loro, per avere oltre 12V corrente continua, e generare parecchi litri di soluzione d'argento, per meglio

dire di argento colloidale.
Colloide significa, in parole povere, che l'argento è in sospensione nell'acqua. Non occorre fare precipitare in quantità l'argento che verrebbe in tal caso perduto.
Non voglio dilungarmi troppo in dissertazioni mediche o chimiche non essendo questa la sede ne io esperto in materia quindi rimando tutto ad una vostra approfondita ricerca su internet dove brulicano i siti, i forum su questa nuova, se così possiamo definirla, tecnica curativa. Una descrizione molto interessante la potete trovare sul sito http://www.naturmedica.com/argentocolloidale.htm.

Figura 1*:* *Bacchette in argento purissimo ed il tappo di gomma per realizzare la sonda*

Ho consigliato la realizzazione del più semplice generatore di corrente per produrre il colloide ma certamente, noi elettronici, non ci fermeremo qui!
Vi propongo la realizzazione di un generatore di corrente costante a più step di tipo professionale con inversione di polarità sulle bacchette di argento, in modo che non si abbia a consumarsi una bacchetta più dell'altra: Con questo circuito potrete realizzare un bicchiere di soluzione di argento in circa una decina di minuti!.
Il circuito ha tre step di regolazione della corrente in modo da avere produzione minima o lenta, media, massima o veloce. Un led bicolore indicherà le differenti polarità della tensione applicata alla sonda. Il circuito è alimentato a tensione di rete tramite un trasformatore a doppio isolamento.

SCHEMA ELETTRICO

Lo schema elettrico di figura 1 può facilmente essere suddiviso in due sezioni principali: la prima relativa al controllo temporizzato di inversione di plarità e la seconda composta dal generatore a corrente costante vero e proprio.
Entrambe le parti vengono alimentate da rete tramite un trasformatore di tensione da 230V a 15V del tipo doppio isolamento a norma CE, quindi la tensione alternata giunge al ponte B1 di piccola potenza ed allo stabilizzatore di tensione con pi greco capacitivo e resistivo con zener a 12V, la tensione in uscita da questo stadio alimenterà la logica di temporizzazione dell'inversione di polarità gestita da un comunissimo C/MOS CD4093B. All'uscita della sua terza porta (NAND) avremo livello opposto che all'ingresso, per cui potremo agevolmente pilotare due relè alternativamente tramite due transistori pilota.
Il circuito provvederà ad eccitare alternativamente, ogni minuto circa, uno dei due relè in modo tale da invertire i poli di

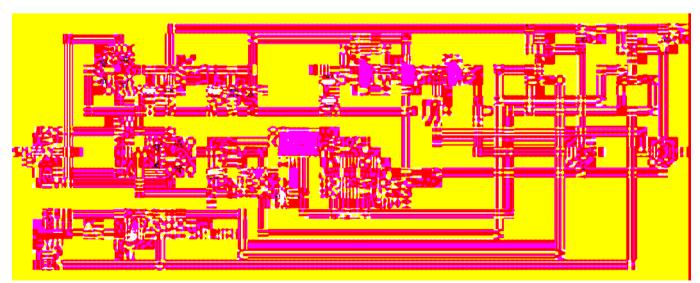



Figura 2*:* *chema elettrico del generatore*

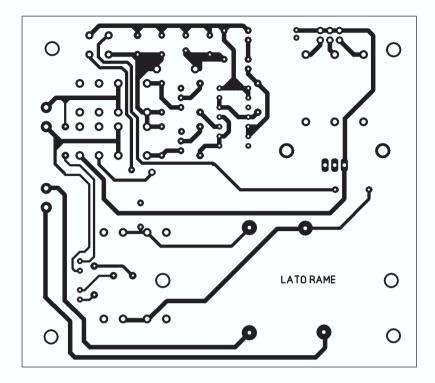



***Figura 3a**: Circuito stampato in scala 1:1 (lato rame)*

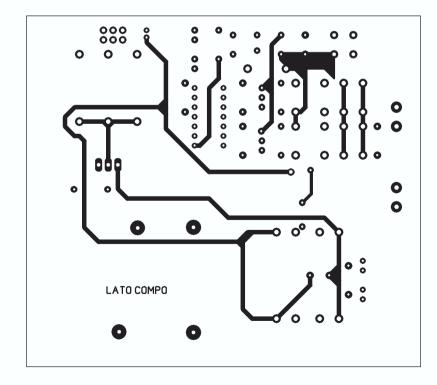



***Figura 3b**: Circuito stampato in scala 1:1 (lato componenti)*

uscita della sonda, invertendo il flusso nel liquido e preservando le astine di argento da differente corrosione tra loro.
Sempre dal secondario del trasformatore T1 preleveremo la tensione per il generatore, utilizzando un ponte composto di quattro diodi 1N5400, livellando il tutto con il condensatore C1.
L'integrato IC2 è impiegato come regolatore di corrente costante a tre step, preimpostabili commutando i resistori R1, R2 e R3 tramite il piccolo dip switch che voi potrete sostituire con un comune commutatore tre posizioni una via. In questo modo avrete tre differenti soglie massime di corrente preimpostabili in modo da ottenere una produzione minima, media o massima di colloide.
L'integrato IC2 è un comune LM317 in contenitore TO220 al quale abbiamo inserito resistore di limitazione tra l'uscita ed il pin di aggiustamento della tensione, in questo modo abbiamo realizzato un generatore di corrente costante, ovvero il circuito modificherà la tensione in uscita a seconda della conduttività dell'acqua e con un massimo di corrente preimpostato e non superabile.
Questo è molto importante perché, specie dopo un certo periodo di trattamento, si potrebbe verificare un aumento di corrente non controllabile e non auspicabile per ottenere una corretta produzione di colloide.
Il led DL2 indica la presenza di tensione al generatore ed il led

DL1, di tipo bicolore, la polarità della tensione alla sonda. Modificando i valori dei resistori di potenza R1, R2 e R3 potrete, a vostro piacere, definire differenti soglie di corrente massima preimpostabili, fino alla massima corrente erogata da IC2 e T1. In questo modo potrete utilizzare il circuito per ottenere cloro dall'acqua salata e realizzare molti processi elettrolitici interessanti.

ISTRUZIONI DI MONTAGGIO

Per questo circuito è stato approntato uno stampato doppia faccia, realizzato con il programma Sprint Layout che tra l'altro permette di esportare il progetto in file Gerber, standard molto apprezzato e utilizzato da tutti i masteristi, ora realizzare un PCB doppia faccia diviene cosa semplice

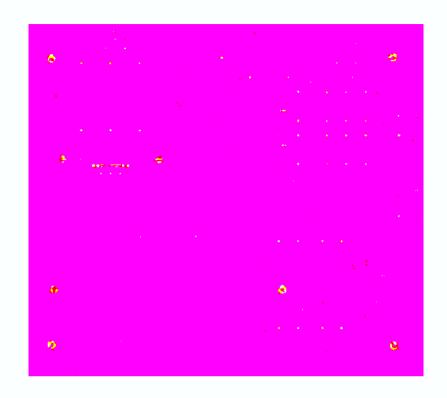



Figura 4: Posizionamento dei componenti

Elenco componenti			
Sigla	**Valore**	**Sigla**	**Valore**
R1	22 Ω 2 W	LD1	Led bicolore due pin 3 mm
R2	15 Ω 2 W	LD2	Led rosso 3 mm
R3	10 Ω 2 W	TR1, Tr2	BC337
R4, R5	4,7 KΩ 1\4 W 5%	B1	Ponte raddrizzatore 50 V 1 A
R6, R9	1 KΩ 1\4 W 5%	DZ1	Diodo Zener 12 V 1 W
R7	120 Ω 1 W	D1÷D4	1N5400
R8	2,2 MΩ 1\4 W 5%	D5, D6	1N4148
C1	2200 µF 25 V elettrolitico	RL1, RL2	Relé doppio scambio miniatura – 3 A bobina 12 V
C2	1 µF 16 V elettrolitico	T1	Trasformatore 230\15 V 1,2 A
C3, C4	22 µF 16 V elettrolitico	1 morsettiera 4 posti passo 5 mm con estrattore e serraggi a vite	
C5, C6	220 µF 25 V elettrolitico	1 doppio selettore molex 2x3 passo 2,54mm	
C7, C8	100 nF 63 V poliestere	1 ponticello singolo passo 2,54 mm	
IC1	CD4093B	1 dissipatore verticale per TO220	

ed attuabile. Sul circuito sono presenti solo due ponticelli, da effettuare con spezzone di reoforo da resistenza, uno vicino ai diodi raddrizzatori, l'altro presso IC2.

Per iniziare il montaggio preferite per primi i resistori quindi condensatori ed integrato, possibilmente con zoccolo, quindi il dissipatore con IC2, i led ed infine il trasformatore ed i morsetti. L'integrato IC2 se utilizzato per produzione di argento colloidale, nel qual caso la corrente per la produzione non è notevole, non scalderà eccessivamente è però necessario per l'utilizzo continuato il dissipatore di calore.
Ora non resta che controllare il circuito, il lavoro svolto, le polarità dei componenti che tassativamente non dovranno essere invertite, dopo l'accurata occhiata al circuito potrete dare tensione. Non appena collegato alla rete si accenderà il led DL2 spia di presenza tensione de il led DL1 brillerà di rosso o di verde: Questo led sarà molto luminoso se la sonda non è immersa in acqua, diminuirà luminosità se immersa e dovrà spegnersi se cortocircuitata. Dopo circa un minuto il led se acceso di verde diverrà rosso o viceversa. Il ciclo si ripeterà finchè il circuito è alimentato. Dopo aver effettuato queste prove sarete sicuri del funzionamento. Non resterà che racchiudere tutto in una scatoletta plastica, del tipo areato, realizzare la connessione per la sonda.

Figura 5: *Sonda immersa nel bicchiere*

REALIZZAZIONE DELLA SONDA

Prendete un semplice tappo di sughero o gomma poi provvedete a fare due fori per inserire le bacchette di argento mantenendole distanziate tra loro di circa 3 cm oppure utilizzate, come in figura 4, una spiaggetta plastica che forerete per infilare le bacchette in argento.
Tanti possono essere i modi di utilizzo e realizzazione della sonda. Si deve però ricordare che più lontane sono le bacchette tra loro e maggiore dovrà essere la tensione di esercizio per generare colloide. Nel prototipo da me realizzato le bacchette sono distanti tra loro circa1 cm e la tensione di esercizio rientra nella finestra tra 6 e 12 Vcc.

La sonda in argento è soggetta ad usura, bilaterale in questo caso, essendoci inversione di polarità. Durante il funzionamento il circuito potrebbe un poco scaldare. Qualora dopo circa una mezzora di trattamento, tempo consigliato, si notasse un deposito nero sul fondo, o abbiamo usato argento non puro o vi è troppo precipitato, ossia la soluzione è troppo satura, diminuire la corrente di trattamento ed il tempo di erogazione.
Qualora invece durante il trattamento una o l'altra delle bacchette o alternativamente tendesse ad ossidare, dovrete abbassare la corrente di trattamento.
Per sospendere l'argento utilizzate acqua distillata per il 75 % e 25 % di acqua naturale poco mineralizzata da tavola. Solo acqua distillata non permette il trattamento.

CONCLUSIONI

Per poter definire concentrazioni di argento in sospensione, consiglio l'acquisto di un conduttivimetro digitale e di un buon libro trattante le applicazioni di colloide d'argento o, se vorrete un poco pazientare, attendete il conduttivimetro che presenterò in un prossimo articolo.

Buon lavoro a tutti.

Figura 6: *Generatore commerciale di argento colloidale*

VITAMINA C
tredicesima parte

STRUTTURA E LEGGIBILITÀ DEL CODICE

di Antonio Di Stefano
a.distefano@farelettronica.com

In questa puntata verranno presi in considerazione alcuni aspetti della programmazione in C che non hanno a che fare direttamente con la sintassi del linguaggio, e forse per questo vengono solitamente poco enfatizzati, ma che invece sono fondamentali per la scrittura di buoni programmi. Ad esempio come va progettato e strutturato un programma completo? Quali accorgimenti bisogna seguire nella scrittura del codice?

Una delle maggiori potenzialità del linguaggio C risiede nella sua grandissima versatilità ed espressività. Esso permette cioè di descrivere il comportamento voluto in molti modi diversi e comunque sempre in maniera molto compatta. Questa caratteristica, di per se positiva, può potenzialmente essere fonte di diversi problemi se non viene ben gestita! Infatti un codice molto compatto, se non ben progettato e scritto può risultare poco leggibile, scarsamente testabile e documentabile, poco portabile, e difficilmente aggiornabile. Al contrario queste caratteristiche dovrebbero sempre essere considerate importanti almeno quanto il corretto funzionamento del programma stesso. Per facilitare questo compito il linguaggio C mette anche a disposizione alcuni strumenti, che gestiti con cura dal programmatore possono assicurare degli ottimi risultati. Di seguito analizzeremo alcuni tra questi strumenti, nonché alcune regole pratiche da seguire per potere scrivere dei buoni programmi.

LEGGIBILITÀ DEL CODICE

Provate a capire a cosa serve il codice C che segue:

```
char *dtb(char n) {
int i; char s[8];
s[8]=0;
for(i=0; i<8; i++)
s[i]=48+(n>>i)&1;
return s;
}
```

Provate ora a leggere questo:

```
/* Conversione decimale -> binario
   Parametri:
   - dec = numero intero (0-255)
   Restituisce:
   - “dec” in binario (stringa) */
char *DecToBin(unsigned char dec)
{
  int i;  /* indice posizione */
  char bin[8];  /* stringa */

  bin[8]=0; /* terminatore stringa */
  for(i=0; i<8; i++)
  {
```

```
    /* scorre n a destra di i posti
       e considera solo il LSB
       (equivale a: n/(2^i) AND 1) */
    bin[i]='0' + (dec>>i)&1;
  }
  return bin;
}
```

Le due routine eseguono esattamente le stesse operazioni, ma nel secondo caso non c'è niente da aggiungere, il codice è autoesplicante.
Cosa rende il secondo codice molto più comprensibile del primo?
In primo luogo i commenti, che come detto in C possono essere costituiti da testo qualsiasi, anche su più line, racchiuso tra /* e */.
È successo a molti, alle prime esperienze con la programmazione di rileggere del codice scritto solo una settimana prima, e trovarlo assolutamente incomprensibile!
Esperienze come queste insegnano quanto sia importante un uso massiccio di commenti nel propri programmi. Un altro accorgimento per rendere il codice più leggibile è quello di utilizzare l'indentazione.
Sicuramente l'avrete notata in tutti i programmi presentati, però essa non è mai stata citata esplicitamente: consiste nell'utilizzare degli spazi (o delle tabulazioni) in modo da allineare verticalmente il codice che appartiene ad uno stesso blocco o contesto. Ad esempio nel programma precedente tutto il codice dentro funzione inizia dopo due spazi, così come quello dentro al ciclo *for* (che si troverà quindi a circa quattro spazi dal bordo della pagina). Questo accorgimento permette di suddividere logicamente e semanticamente il codice, e quindi di poterlo seguire e comprendere più facilmente.
L'indentazione è anche molto utile per rendersi conto in maniera veloce di quante parentesi graffe occorre chiudere alla fine di una routine.
Un ulteriore (e forse decisivo) aiuto alla leggibilità del codice è dato dall'utilizzo di nomi significativi per le variabili e le funzioni. È possibile utilizzare anche nomi piuttosto lunghi, composti anche da lettere maiuscole (il C è case sensitive!) o contenenti il segno "_" (underscore) per evidenziare diverse parole all'interno dello stesso nome: è molto più espressivo un nome come *vettore_coefficienti[]* che non *v[]*… Questa idea si può estendere al punto da seguire delle precise convenzioni, come quella di utilizzare la prima lettera del nome di una variabile per indicarne il tipo di dato da esso rappresentato, ad esempio per rappresentare una variabile di tipo float la si chiamerà *fLunghezza*, mentre una variabile intera potrebbe chiamarsi *iContatore*…
Ovviamente applicare scrupolosamente tutti questi accorgimenti comporta un leggero aumento del tempo necessario alla scrittura del codice, però i vantaggi che ne derivano sono talmente grandi da giustificare sicuramente il maggiore sforzo.

STRUTTURA DEI PROGRAMMI

Così come è bene rispettare certe regole nella scrittura delle singole linee di codice e delle routine, è anche opportuno considerare alcuni accorgimenti nell'organizzazione e nella scrittura dell'intero programma. In questo caso gli obbiettivi a cui mirare sono, oltre alla leggibilità, anche la riutilizzabilità del codice, la facilità di modifica o aggiornamento e la portabilità (indipendenza dall'hardware).
Prima di vedere in dettaglio come si possono raggiungere questi risultati è il caso di introdurre i file header. Si tratta di file che hanno lo stesso nome dei file di programma, ma hanno estensione ".h" anziché ".c".
I file *header* sono quelli che fino ad ora abbiamo richiamato utilizzando la direttiva *#include*, per esempio nel caso delle librerie. Mentre i file di programma contengono il codice vero e proprio delle routine, negli header sono presenti soltanto i prototipi delle funzioni e la dichiarazione di macro e costanti. Per prototipo di una funzione si intende una dichiarazione contenente il nome della funzione ed i tipi da essa utilizzati in ingresso ed uscita. L'uso dei prototipi non è strettamente necessario, ma permette di trovare facilmente degli errori poco visibili nella scrittura del codice. Infatti se nel programma viene richiamata una funzione utilizzando dei parametri di un tipo diverso da quello dichiarato nel prototipo, il compilatore darà un avvertimento *(warning)* o addirittura un errore.
L'utilizzo dei file *header* ha lo scopo di separare la dichiarazione delle funzioni (cioè la loro interfac-

cia) dal codice vero e proprio. Conoscendo l'interfaccia di una funzione (cioè le variabili in ingresso, quelle in uscita ed i rispettivi tipi) è possibile utilizzarla senza bisogno di conoscere il suo funzionamento interno e la sua implementazione. Questo approccio viene chiamato *information hiding*, proprio perché tende a nascondere le parti più complesse e quindi meno leggibili del programma.
A prima vista forse questo meccanismo potrebbe sembrare un po' complicato, ma ripensandoci esso è lo stesso che utilizziamo ogni volta che impieghiamo una funzione di libreria. Anche in quel caso richiamiamo un file header, ed utilizziamo delle funzioni di cui conosciamo soltanto l'interfaccia, ma non la reale implementazione.
Dovrebbe essere chiaro a questo punto che gli *header* sono utili sia per aumentare la leggibilità, sia soprattutto per favorire la riutilizzabilità del codice. Infatti è possibile, ed in molti casi conveniente, creare delle proprie librerie di funzioni, scrivendo il codice in un apposito file, e creando il rispettivo *header*. Ogni volta che si dovranno utilizzare quelle funzioni in un programma sarà sufficiente solamente includere il file *header*!
Ecco un piccolo esempio relativo al codice di un ipotetica libreria che implementa un paio di funzioni statistiche.
Il file *stat.c* conterrà il seguente codice:

```
#include <stat.h>

/* Funzione Sum */
int Sum(char *vettore, int num_elem)
{
  int i, somma=0;

  for(i=0; i<num_elem; i++)
    somma += vettore[i];
  return somma;
}

/* Funzione Med  */
int Med(char *vettore, int num_elem)
{
  int i, media=0;

  for(i=0; i<num_elem; i++)
    media += vettore[i];
  return media/num_elem;
}
```

Il file stat.h invece conterrà le seguenti dichiarazioni:

```
/* *** Sum ***
   Restituisce la somma degli elementi
   dell'array di byte "vettore" */
int Sum(char *vettore, int num_elem);

/* *** Med ***
   Restituisce la media degli elementi
   dell'array di byte "vettore" */
int Med(char *vettore, int num_elem);
```

Per utilizzare le due funzioni nei propri programmi basta includere il file *stat.h*! (I due file devono risiedere in una directory nota al compilatore, altrimenti occorre specificarla nella direttiva include).
Un'osservazione banale: il file *stat.c* non ha un main proprio perché il codice non deve funzionare da solo, ma deve essere sempre richiamato da un altro programma.

Negli *header* di solito si dichiarano anche le costanti ed i tipi definiti dall'utente, che poi verranno usate nel codice. Il vantaggio in questo caso risiede nel fatto che se il valore di queste costanti deve essere cambiato, è sufficiente farlo una sola volta senza bisogno di accedere al codice. Questo offre un modo per parametrizzare i propri programmi, e poterli personalizzare a seconda delle esigenze. Alcune costanti tipicamente dichiarate in questo modo sono quelle legate alle dimensioni massime degli *array*. Anche per i tipi vale lo stesso discorso. Nell'esempio precedente per esempio avremmo potuto utilizzare il tipo *INTERO_CORTO ed INTERO_LUNGO* al posto di *char* ed *int*, e definirli così nel file *header*:

```
typedef char INTERO_CORTO;
typedef int INTERO_LUNGO;
```

In questo modo le nostre funzioni avrebbero potuto utilizzare anche vettori di *long* o *float*, semplicemente ritoccando queste due righe dell'*header*!

Suddividere un programma in diversi moduli (cioè

diversi file .c ed i rispettivi .h) è sicuramente una buona pratica, infatti essa permette in primo luogo di partizionare il programma in parti più piccole e semplici da sviluppare (tecnica nota come progettazione "top-down" o approccio "divide et impera"), ed in secondo luogo favorisce la riutilizzabilità del codice. Non solo, in questo modo è anche possibile ottenere maggiore portabilità del codice, isolando in moduli appositi tutte le funzioni dipendenti dall'hardware. Se dovesse essere necessario utilizzare lo stesso programma su un hardware diverso, sarebbe sufficiente ritoccare soltanto i moduli che vi accedono direttamente, lasciando inalterato tutto il resto!

ESEMPIO PRATICO

Di seguito è riportato un esempio di programma completo che segue tutti gli accorgimenti citati. Nonostante la sua estrema semplicità (dovuta ad esigenze "didattiche"), esso ha una struttura completa e molto simile a quella che potrebbe avere un programma molto più grande e complesso. Il programma dovrebbe gestire un sistema a microcontrollore che ha lo scopo di tenere sotto controllo i dati provenienti da alcuni sensori, controllando che essi si mantengano entro dei valori limite stabiliti. La condizione di normalità o di avaria è visualizzata su un display LCD alfanumerico.
Il programma è costituito da 3 moduli: il modulo "main.c", che contiene le routine principale del programma, il modulo "sensori.c" che si occupa della lettura dei dati dai sensori, ed il modulo "display.c" che invece ingloba le funzioni per il pilotaggio del display. Ciascuno dei moduli ha il proprio *header*. Va notato che questa struttura rende il programma abbastanza indipendente dall'hardware, infatti esso potrebbe perfino funzionare su un PC, modificando soltanto le funzioni dei moduli *sensori.c* e *display.c* in modo che utilizzino ad esempio *scanf* e *printf* per leggere i dati e visualizzare l'output.
Anche i parametri di funzionamento (limiti impostati e coefficienti di conversione) risultano facilmente modificabili dagli *header*.

File "main.c"

```
/* ******************************
   *    PROGRAMMA DI CONTROLLO    *
   *      Modulo principale       *
   ****************************** */
#include "main.h"

main()
{
  int valore, err_temp, err_press;
  /* ciclo infinito */
  while(1)
  {
    /* Controllo temp. acqua */
    err_temp=0;
    valore=TempAcqua();
    if (valore>MAX_TEMP_ACQUA)
      err_temp=1;
    /* Controllo press. olio */
    err_press=0;
    valore=PressOlio();
    if ((valore<MIN_PRESS_OLIO)||
      (valore>MAX_PRESS_OLIO))
      err_press=1;
    if ((err_temp==1)||(err_press==1))
    {
      Scrivi("Rilavata anomalia!");
      Buzzer();
    }
    else
      Scrivi("Funzionamento normale");
  }
}
```

File "main.h"

```
/* ******************************
   *    PROGRAMMA DI CONTROLLO    *
   *  Header modulo principale    *
   ****************************** */
#include "sensori.h"
#include "display.h"

/* --- Definizione parametri --- */
/* Massima temperatura acqua */
#define MAX_TEMP_ACQUA 98
/* Minima pressione olio */
#define MIN_PRESS_OLIO 8
/* Massima pressione olio */
#define MAX_PRESS_OLIO 21
```

File "sensori.c"

```
/* **************************
   * Modulo gestione sensori *
   *        - codice -        *
   ************************** */
#include "sensori.h"

/* -------------------------- */
int TempAcqua(void)
{
  int i;
  /* Istruzioni dipendenti
     dall'hardware */
  return i*COEFF_TEMP;
}
/* -------------------------- */
int PressOlio(void)
{
   int i;
  /* Istruzioni dipendenti
     dall'hardware */
   return i*COEFF_PRESS;
}
```

File "sensori.h"

```
/* **************************
   * Modulo gestione sensori *
   *        - header -        *
   ************************** */

/* -- Definizioni parametri -- */

/* Coeff. di convers. temp. */
#define COEFF_TEMP 1.5
/* Coeff. di convers. press. */
#define COEFF_PRESS 3.2

/* - Prototipi delle funzioni - */

/* Legge il valore del sensore
   della temperatura dell'acqua */
int TempAcqua(void);
/* Legge il valore del sensore
   della pressione dell'olio */
int PressOlio(void);
```

File "display.c"

```
/* **************************
   * Modulo gestione display *
   *        - codice -        *
   ************************** */
#include "display.h"

/* -------------------------- */
void Scrivi(char *testo)
{
  /* Istruzioni dipendenti
     dall'hardware */
}
/* -------------------------- */
void Buzzer(void)
{
  /* Istruzioni dipendenti
     dall'hardware */
}
```

File "display.h"

```
/* **************************
   * Modulo gestione display *
   *        - header -        *
   ************************** */
#include <stdio.h>

/* - Prototipi delle funzioni - */
/* Scrive un rigo di
   testo sul display */
void Scrivi(char *testo);
/* Emette un beep */
void Buzzer(void);
```

CONCLUSIONI

Sono sicuro che acquisita un po' di esperienza nell'uso del linguaggio, apprezzerete ancora di più gli argomenti trattati in questa puntata, e vi renderete conto gradualmente di quanto tempo e fatica può fare risparmiare un programma ben scritto fin dall'inizio.

Nella prossima puntata cambieremo decisamente argomento, e ci occuperemo invece dell'ambiente di sviluppo e del compilatore dedicato ad un popolare microcontrollore.

SMARTCARD
sesta parte

LA SIM: UN ESEMPIO DI SMARTCARD A MICROPROCESSORE

di Giuseppe Modugno
gppe.modugno@libero.it

In questo articolo, prenderemo tra le mani un tipo molto diffuso di smartcard a microprocessore, le SIM comunemente usate all'interno dei telefonini GSM. Verrà descritto il funzionamento di tali smartcard all'interno del sistema di telefonia mobile GSM: funzionalità, organizzazione interna dei dati, comandi, ecc.

RIPRENDIAMO LA NOSTRA PARTITA A CARTE

Cosa ci siamo detti negli articoli precedenti

Per chi non avesse avuto la possibilità di leggerci sin dall'inizio del nostro tutorial, facciamo una breve panoramica dei contenuti degli articoli precedenti.
Nella prima parte, abbiamo introdotto il termine smartcard. Abbiamo effettuato alcune classificazioni fondamentali (tra cui, le smartcard a memoria e a microprocessore) e presentato gli standard internazionali di riferimento, formulati dall'ISO, che le smartcard più diffuse devono seguire. Nel secondo e terzo articolo, sono state approfondite rispettivamente le smartcard a memoria (in particolare quelle basate sul chip SLE4442) e le smartcard a microprocessore (standard e "meno standard"). Nella quarta parte, abbiamo finalmente messo mano al saldatore, realizzando un economico lettore universale di smartcard, denominato UniReader, compatibile sia con le smartcard a memoria che con quelle a microprocessore. Infine, nel precedente articolo (quinta parte), abbiamo utilizzato le smartcard a memoria in alcune applicazioni pratiche, realizzando un firmware ad hoc per l'UniReader (sia in versione stand-alone, sia in versione slave).
In questa parte e nella successiva, descriveremo nei dettagli una smartcard che, con molta probabilità, è già in nostro possesso, magari in più esemplari: la SIM dei telefoni GSM. Il lettore che ci segue sin dall'inizio sa bene che questo oggetto non è altro che una smartcard a microprocessore che può essere gestita alla stessa stregua delle altre smartcard. Descriveremo gli standard a cui le SIM fanno riferimento, le funzioni nel sistema GSM, i comandi che permettono la gestione dei dati interni, l'organizzazione delle varie informazioni utilizzate dal sistema.

Breve panoramica sulle smartcard a microprocessore

Consiglio al lettore che non

avesse letto il nostro tutorial di recuperare almeno la terza parte che contiene una descrizione dettagliata sulle smartcard a microprocessore (oltre alla prima parte che considero propedeutica). In questo breve paragrafo farò solo una veloce panoramica su questo tipo di carte per "rinfrescare" la memoria.
Le smartcard a microprocessore sono delle normali smartcard che integrano internamente, a differenza delle semplici smartcard a memoria, un vero e proprio microprocessore, una memoria volatile e veloce (RAM), una memoria permanente più lenta (EEPROM o FLASH) ed un porta seriale. Possiamo paragonare una di queste smartcard ad un vero e proprio computer miniaturizzato, dotato di una CPU, di una memoria e di una sola periferica di I/O, la porta seriale ISO7816 (da non confondere con la classica RS232 presente sui PC). Attraverso la porta seriale la carta comunica con il mondo esterno (dispositivo di interfaccia) ricevendo e rispondendo ai comandi più svariati. Questo collegamento seriale utilizza un preciso protocollo di trasferimento dei dati, tra cui il più diffuso e semplice è quello denominato "T=0", descritto nella parte terza dello standard ISO7816. Il microprocessore, insieme alla memoria RAM, effettua tutte le elaborazioni necessarie per la gestione dei dati memorizzati nella carta (lettura e scrittura) e per la esecuzione degli eventuali algoritmi di cifratura necessari per la segretezza delle transazioni. La memoria non volatile è naturalmente usata per la memorizzazione dei dati dell'utente o dell'applicazione anche ad alimentazione assente: numeri seriali, dati anagrafici, chiavi crittografiche, ecc.
Come abbiamo visto nella terza parte, lo standard ISO7816-3 definisce un preciso protocollo per l'invio dei comandi. Questi sono formati da una intestazio-

ne di cinque byte, denominati **CLA INS P1 P2 LEN** e trasferiti sempre dal lettore verso la smartcard: in altre parole, la smartcard non può mai decidere di iniziare un trasferimento dati di sua iniziativa, ma può solo rispondere ai comandi ricevuti dal dispositivo di interfaccia. Subito dopo l'invio di questo comando, la smartcard risponde con un procedure byte che indica al lettore come gestire correttamente la tensione di programmazione Vpp: nella maggior parte dei casi, quando la tensione Vpp non viene usata, il procedure byte coincide con il byte **INS**. Dopo l'invio di questo byte, la smartcard può rispondere con un pacchetto di byte oppure il lettore può inviare ulteriori dati alla smartcard: il verso del trasferimento dei dati dipende dal tipo di comando inviato (fondamentalmente dalla coppia dei byte **CLA** e **INS**). È necessario che sia il lettore che la smartcard abbiano la stessa conoscenza a priori del significato dei vari comandi e, quindi, della direzione di trasferimento dei byte. Se il comando prevede un trasferimento dalla carta verso il lettore, **LEN**=0 indica l'invio di 256 byte; viceversa, se il comando indica un trasferimento dal lettore verso la carta, **LEN**=0 indica che non ci sarà alcun trasferimento di byte.

ISO7816 PARTE 4

Nell'articolo sulle smartcard a microprocessore abbiamo fatto riferimento allo standard ISO7816 parte 3. Come abbiamo visto, questo standard si limita a definire, per le smartcard asincrone (cioè quelle a microprocessore), la struttura dell'ATR ed il protocollo T=0 per il trasferimento di dati e comandi nelle due direzioni. Non entra nel merito dell'organizzazione interna dei dati né su come devono essere gestiti all'interfaccia lettore-smartcard. In altre parole, limitandosi alla terza parte dell'ISO7816, è possibile creare smartcard e lettori incompatibili tra loro. Basti pensare, per esempio, ad una diversa codifica dei comandi (**CLA** e/o **INS**).

È per questo motivo che è stata pubblicata una quarta parte dello standard ISO7816 che definisce una organizzazione logica dei dati alla sezione smartcard-lettore.

L'implementazione interna, cioè l'allocazione dei dati e la loro codifica nella memoria interna della smartcard, rimane ancora non definita e può essere liberamente ideata dallo sviluppatore dell'applicazione.

La cosa importante è che questi dati possano "apparire" nel modo standard al lettore. Poiché le SIM GSM seguono anche questo standard, è utile approfondirne almeno alcuni aspetti principali.

Organizzazione logica dei dati: i file

Così come in un PC, anche nelle smartcard ISO7816-4 i dati vengono organizzati in una struttura gerarchica di file. Lo standard definisce i *Dedicated File* (DF) e gli *Elementary File* (EF): i primi sono una sorta di cartelle, spesso chiamati *directory*, mentre i secondi assomigliano ai file veri e propri di un PC. Il DF principale viene chiamato *Master File* (MF) e rappresenta la radice della struttura gerarchica ad albero. L'MF può contenere altri DF oppure EF. A loro volta, i DF di primo livello possono contenere altri DF oppure EF e così via. Ogni file è individuato da un identificatore *(file identifier)* di due byte. L'MF è sempre identificato da '3F00', il valore 'FFFF' è riservato per usi futuri, mentre il valore '3FFF' è riservato per la selezione di un file mediante percorso relativo (vedi oltre).

Come nel vecchio DOS, anche per le smartcard c'è una directory (DF) corrente; inoltre, c'è anche un file (EF) corrente. All'inizio della sessione (dopo l'ATR) l'MF è la directory corrente, mentre non c'è alcun file selezionato. Successivamente, è possibile selezionare un diverso file (DF o EF) mediante il comando SELECT FILE, attraverso le seguenti modalità:

- referenziando il suo identificatore a 2 byte;
- referenziando il suo percorso assoluto (a partire dall'MF) o relativo (a partire dal DF corrente) citando gli identificatori di tutti i file intermedi;
- referenziando l'identificatore breve (short file identifier) del file (se implementato);
- referenziando il nome univoco (da 1 a 16 byte) associato al DF che si vuole selezionare (se implementato).

Lo standard prevede due tipi di file: transparent file e record file. I primi sono visti come una sequenza di byte a partire dal-

l'indirizzo 0 fino all'indirizzo n-1, dove n è la dimensione dei dati del file. Per esempio, è possibile memorizzare un numero di serie o una chiave crittografica in un file di questo tipo. I file basati su record, invece, possono essere letti e scritti solo a gruppi di n byte, dove n è generalmente costante e dipende dal file. Per esempio, è possibile utilizzare un file di record per memorizzare una rubrica di numeri di telefono: ogni nominativo occuperà un intero record. Esistono tre tipi di file basati su record: *linear fixed file*, *linear variable file* e *cyclic file*. I linear fixed file sono una sequenza di record a lunghezza fissa a cui si può accedere mediante un numero di record. I linear variable file sono identici ai fixed con la differenza che la dimensione di ogni record può essere variabile (non sono molto usati nelle applicazioni comuni). Infine, i cyclic file sono sempre basati su record a lunghezza fissa a cui si può accedere ciclicamente: l'ultimo record aggiunto (in ordine di tempo) sarà il primo ad essere letto.

Nei transparent file è possibile accedere ai dati mediante un offset relativo al primo byte (indirizzo 0). Nei file strutturati a record, invece, si può accedere ai dati mediante un numero di record (a partire da 1) oppure mediante un puntatore ad un record corrente.

Il software di gestione della smartcard può conoscere il tipo di file e tutti i suoi attributi (dimensione, numero di record, lunghezza dei record, ecc.) tramite la risposta al comando SELECT FILE che rappresenta il *File Control Information* (FCI). È una sequenza di byte che permette di risalire al tipo di file ed a tutti i suoi parametri.

Ad ogni file è associato un meccanismo di sicurezza che protegge i dati da operazioni di lettura, scrittura, ecc. Per esempio, se un file è protetto in lettura, sarà necessario inserire correttamente un codice segreto (PIN) per abilitare tale operazione. Anche il tipo di protezione è indicato nell'FCI.

In realtà, lo standard ISO7816-4 è molto complicato e generico e descrive una serie di comandi per l'accesso ai dati memorizzati nei file della smartcard: comandi di lettura e scrittura, di autenticazione, di selezione dei file, ecc. Nella maggior parte dei casi pratici, però, sono implementati solo una parte dei comandi e delle funzionalità descritte nello standard. Per questo motivo, non approfondiamo ulteriormente il contenuto di queste direttive e rivolgiamo subito la nostra attenzione alle SIM.

SUBSCRIBER IDENTITY MODULE

Cos'è una SIM

La SIM (Subscriber Identity Module) è una smartcard utilizzata nel ben noto sistema di telefonia cellulare GSM per scopi di autenticazione dell'utente e sicurezza delle conversazioni durante le comunicazioni con gli apparati di rete del proprio gestore di servizi. Gli innumerevoli standard che formano il sistema GSM sono formalizzati dall'ETSI e disponibili gratuitamente sul suo sito, previa registrazione. I più importanti documenti relativi alle SIM e al loro funzionamento, sono:

- GSM 02.17 "Subscriber Identity Modules, Functional Characteristics", che descrive le funzionalità (a livello logico) ed il ruolo di una SIM all'interno del sistema GSM;
- TS 100 977 (3GPP TS 11.11) "Specification of the SIM-ME Interface", che descrive l'interfaccia (ad un livello tecnico) tra la SIM ed il telefono cellulare.

Una SIM è una smartcard (o IC-Card, utilizzando i termini standard) che individua un cliente, o un abbonamento, di un determinato provider. La responsabilità della gestione della carta è del provider che la fornisce al cliente sottoscrittore di un qualsiasi abbonamento.

Il sistema GSM ha introdotto per la prima volta, nell'ambito della telefonia mobile, l'idea di separare logicamente (e fisicamente) il telefono cellulare dall'utente e dai suoi dati. Usando la terminologia degli standard, il telefonino viene definito *Mobile Equipment* (ME), mentre il telefonino dotato di SIM è definito *Mobile Station* (MS). L'ME è solo l'apparato elettronico che permette di effettuare e ricevere telefonate sfruttando la rete GSM, ma *ha bisogno della SIM* per acquisire i permessi e autenticare il cliente presso il gestore telefonico (solo le telefonate d'urgenza possono essere effettuate senza una SIM valida inserita nell'ME), oppure per deci-

frare il traffico dati (voce, SMS, ecc.) da e verso la rete.
Sono previsti due tipi di smartcard a livello fisico: le IC-Card, che hanno le stesse dimensioni delle smartcard ISO7816 (tipo carta di credito) e le Plug-In SIM che hanno una dimensione ridotta (25x15 mm). Le prime possono normalmente essere estratte dal telefonino in modo molto semplice e possono essere delle carte multiapplicazioni di cui il GSM è solo una di queste. Il secondo tipo di carta, molto più piccolo delle IC-Card, è stato introdotto per permettere la miniaturizzazione dei telefonini e, normalmente, è semipermanente. A parte la dimensione fisica, elettricamente e logicamente i due tipi di SIM sono identici. Faccio notare che in commercio esistono degli adattatori che permettono di trasformare una Plug-In SIM in una IC-Card. Consiglio al lettore che volesse sperimentare con le SIM mediante l'UniReader, così come descritto nel prossimo articolo, di procurarsi questo adattatore, dal costo irrisorio, presso tabaccherie o negozi per telefonia, in quanto il nostro lettore prevede solo le IC-Card. In alternativa, è possibile costruirsene uno facendo del semplice bricolage, ritagliando opportunamente un cartoncino (o plastica) rigido dello spessore di una SIM e incollando due fogli da entrambe le parti, in modo che rimanga esternamente una finestrella per l'inserimento della SIM e per i suoi contatti.

Autenticazione

Il sistema di autenticazione in una qualsiasi rete di telecomunicazioni è estremamente importante poiché potrebbe inficiarne il reale utilizzo e la sua diffusione. Basti pensare alla precedente tecnologia di telefonia cellulare ETACS e ai numerosissimi casi di clonazione dell'identità dell'utente. Il sistema GSM è completamente digitale e permette, quindi, di utilizzare tecniche più efficaci di autenticazione (crittografia). In questo paragrafo farò un piccolo accenno al meccanismo utilizzato dalla rete GSM per riconoscere un cliente che ha diritto di usufruire dei servizi messi a disposizione dall'infrastruttura.
Ogni SIM rilasciata al cliente dal gestore, proprietario di una rete di telefonia cellulare GSM (PLMN, *Public Land Mobile Network*), è individuata univocamente da un numero, chiamato IMSI (*International Mobile Subscriber Identity*) formato al più da 15 cifre decimali: MCC (*Mobile Country Code*) di 3 cifre, che individua la nazione dell'operatore (222 per l'Italia); MNC (*Mobile Network Code*), che individua l'operatore all'interno della nazione (10 per Vodafone in Italia); MSIN (*Mobile Station Identification Number*) al più di 10 cifre, che rappresenta il numero seriale creato dal provider e che individua univocamente un suo cliente. Non è un numero segreto, tant'è che è possibile leggerlo, come faremo nel prossimo articolo, direttamente dalla SIM selezionando il file relativo.
Associato all'IMSI c'è un altro identificatore di 16 byte, denominato Ki, che individua univocamente il cliente. Il numero Ki è memorizzato segretamente all'interno della SIM, da cui è impossibile leggerlo e all'interno degli apparati di rete del gestore. Infine, la SIM ha la possibilità di eseguire un algoritmo crittografico (definito A3) che permette di trasformare un numero casuale di 16 byte (RAND) in un numero di 4 byte (SRES), utilizzando come chiave la Ki. In figura 1 è mostrato il flusso di informazioni tra la rete GSM e la SIM del cliente (tramite l'ME) durante la fase di autenticazione alla

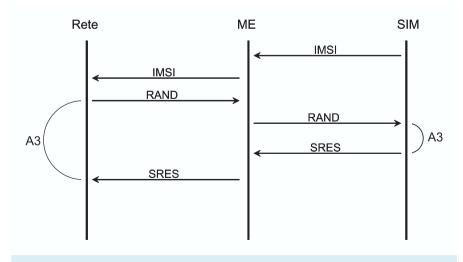



Figura 1: *Procedura di autenticazione alla rete GSM*

rete del gestore.
All'accensione (o quando è necessaria l'autenticazione), l'ME richiede l'IMSI alla SIM e lo invia alla rete dell'operatore mediante l'interfaccia radio. Gli apparati di rete controllano l'esistenza dell'IMSI all'interno di un proprio database e, in caso affermativo, trasmettono alla SIM (tramite l'ME) un numero casuale di 16 byte denominato RAND. Indipendentemente, la SIM e la rete calcolano il numero di 4 byte SRES a partire dal numero casuale RAND e dalla chiave crittografica segreta Ki, utilizzando l'algoritmo *comune* A3. A questo punto, l'ME richiede alla SIM il valore SRES calcolato e lo trasmette al gestore che può controllare la coincidenza tra l'SRES calcolato dalla SIM e quello calcolato dagli apparati di rete. Se i due numeri sono uguali, vuol dire che la SIM è autentica (conosce sia l'algoritmo A3 che la chiave segreta Ki) e può essere abilitata ai servizi di rete. Faccio notare che lo standard GSM non definisce i dettagli dell'algoritmo A3 ma solo l'input (la chiave Ki di 16 byte ed il numero RAND di 16 byte) e l'output (il numero di 4 byte SRES); l'operatore telefonico può utilizzare un qualsiasi algoritmo e, possibilmente, tenerlo segreto.
Questo sistema di autenticazione è molto efficace, poiché si basa sulla segretezza della chiave Ki memorizzata nel database del gestore e nella SIM del cellulare. Essendo la SIM una smartcard a microprocessore, è possibile effettuare elaborazioni (come l'esecuzione dell'algoritmo A3) e rendere invisibli i dati (come la Ki), rendendo il sistema molto più sicuro.
In realtà, il meccanismo descritto in precedenza è semplificato rispetto al caso reale. Per esempio, non è stata presa in considerazione la possibilità che l'IMSI ricevuto dalla rete appartenga ad un altro provider, magari di un'altra nazione. In questo caso, è necessario decidere se è possibile effettuare il roaming consultando le infrastrutture del provider del cliente, mediante uno scambio di informazioni ben più complesso. Accenno solo al fatto che l'autenticazione alla rete avviene di continuo durante l'operatività dell'ME, sfruttando spesso un numero seriale temporaneo (TMSI) piuttosto che l'IMSI per aumentare l'efficienza del sistema.

Segretezza

Oltre al classico problema di autenticazione, una rete di telecomunicazioni deve anche garantire la segretezza delle informazioni scambiate. Ciò è ancor più importante nelle reti di telefonia mobile, poiché il mezzo trasmissivo, cioè la tratta radio tra l'ME e la stazione base (comunemente detta cella o, più tecnicamente, BTS) è "di dominio pubblico". Anche in questo caso, la tecnologia digitale ci permette di utilizzare tecniche di crittografia per cifrare le informazioni in transito e renderle illeggibili ad eventuali "ascoltatori non autorizzati". Nel vecchio sistema di telefonia mobile analogico ETACS era possibile, con appositi strumenti, demodulare il segnale trasmesso in chiaro (per esempio, mediante un analizzatore di spettro evoluto) ed ascoltare le telefonate in corso.
Il sistema GSM, invece, prevede la cifratura di tutti i messaggi scambiati tra l'MS (il telefonino) e la BTS (l'antenna più vicina con cui l'MS comunica) mediante un algoritmo standardizzato ed implementato in ogni cellulare GSM e denominato A5. Durante la comunicazione tra MS e BTS, i dati vengono cifrati mediante l'algoritmo A5, utilizzando una chiave crittografica temporanea di 8 byte, memorizzata nella SIM del cellulare e denominata Kc.
Quello che normalmente accade è che la SIM esegue sia l'algoritmo A3, descritto precedentemente ed utilizzato in fase di autenticazione per la generazione del numero SRES, sia un altro algoritmo, detto A8, per la generazione della chiave di decifrazione dei messaggi Kc. Spesso, la coppia degli algoritmi A3 e A8 vengono implementati come un unico algoritmo, comunemente detto A38, all'interno della SIM, in modo che vengano generati automaticamente sia il numero SRES, sia la chiave Kc. Tutte le informazioni che il telefonino scambia con l'antenna sono codificate con tale chiave. Essa viene cambiata numerose volte durante una sessione (per esempio, una chiamata telefonica) per evitare che si possa risalire alla chiave Kc, intercettando una lunga sequenza di dati codificati con lo stesso codice.

Struttura logica di una SIM

Le SIM GSM sono delle smart-

card a microprocessore che seguono gli standard ISO7816-3 e ISO7816-4. Per questo motivo, i dati sono organizzati logicamente in file, così come descritto in precedenza. Come al solito, esiste un MF con identificatore '3F00' e numerosi DF ed EF figli. Nelle SIM, i DF di primo livello (figli diretti dell'MF) sono identificati da "7Fxx", mentre i DF di secondo livello sono identificati da "5Fxx". Gli EF contenuti direttamente nell'MF sono identificati da '2Fxx', quelli contenuti nei DF di primo livello sono identificati da "6Fxx", infine quelli contenuti nei DF di secondo livello da '4Fxx'. In altre parole, il primo byte dell'identificatore permette di risalire alla posizione (livello) del file all'interno della struttura gerarchica. Nell'assegnazione degli identificatori è necessario che nessun file abbia lo stesso identificatore di un altro contenuto nella stessa directory (DF), così come non possono esserci due file con lo stesso nome in una stessa cartella di un PC.

Ad un EF è associato un header, che indica il tipo di file ed i suoi attributi ed un corpo dati di dimensione variabile. Un DF, invece, ha solo funzioni di contenitore (directory), quindi ha solo un header senza dati. Per poter accedere correttamente ai dati contenuti in un file, è necessario leggere e interpretare l'header associato.

In una SIM sono previsti tre tipi di EF:

- *transparent file*: una sequenza di byte a cui si può accedere mediante un offset relativo (indirizzo), rispetto al primo byte del file (indirizzo 0);
- *linear fixed file*: una sequenza di record a lunghezza fissa a cui è possibile accedere mediante un numero di record (a partire da 1), un puntatore o una ricerca;
- *cyclic file*: una sequenza ciclica di record a lunghezza fissa in cui esiste un collegamento tra il primo record e l'ultimo record, in modo da memorizzare dati in ordine cronologico.

Figura 2: *Possibile struttura di file in una SIM*

Selezione dei file

Così come descritto nello standard ISO7816-4, esiste un comando apposito per la selezione dei file in una SIM, definito SELECT FILE. Dopo l'ATR, viene automaticamente selezionata la directory MF.

Selezionando un DF (o di nuovo l'MF), viene modificata la directory corrente. Dopo questa operazione, non c'è alcun EF corrente. Selezionando un EF, viene impostato come DF corrente la directory padre dell'EF selezionato. Ovviamente, l'EF corrente risulta quello appena selezionato. In qualsiasi momento, in funzione dell'ultimo file selezionato, è possibile selezionare:

1. un qualsiasi file contenuto nella directory corrente;
2. un qualsiasi DF contenuto nel DF padre della directory corrente;
3. il padre della directory corrente;
4. il DF corrente;
5. il Master File.

Per rendere più chiaro il meccanismo, più semplice da capire che da spiegare, consideriamo l'esempio in figura 2, dove è mostrato una possibile struttura gerarchica di file memorizzata in una SIM (l'esempio è identico a quello presente sullo standard).

In Tabella 1 sono mostrati tutti i file che è possibile selezionare, mediante un singolo comando SELECT FILE, a partire da un altro file corrente. Per esempio, se il file corrente è l'EF2, sarà possibile selezionare MF (5), DF1 (4), DF2 (2), DF3 (1). Tra parentesi è indicata la regola

relativa che permette la selezione di quel file.

Anche se non è stato messo in evidenza, è sempre possibile selezionare il file corrente, cioè l'ultimo file selezionato.

Modalità d'accesso ai file

Ogni EF in una SIM ha una modalità d'accesso relativo ad ogni comando (lettura o ricerca, scrittura, abilitazione, ecc). Tale modalità assume un valore intero da 0 a 15, secondo la Tabella 2.

I due codici segreti CHV1 e CHV2 rappresentano dei meccanismi di sicurezza che permettono solo all'utente, che conosce tali codici, di accedere alle informazioni più sensibili della SIM (rubrica, SMS, IMSI, impostazioni, ecc). Il CHV1 (*Card Holder Verification* 1) è il PIN (vecchia terminologia) formato da un minimo di 4 ad un massimo di 8 cifre decimali, il CHV2 (*Card Holder Verification* 2) è un altro codice segreto che può essere utilizzato per altri scopi

Ultimo file selezionato	Selezioni valide
MF	DF1, DF2, EF1
DF1	MF, DF2, DF3, EF2
DF2	MF, DF1, EF3, EF4
DF3	MF, DF1, EF5
EF1	MF, DF1, DF2
EF2	MF, DF1, DF2, DF3
EF3	MF, DF1, DF2, EF4
EF4	MF, DF1, DF2, EF3
EF5	MF, DF1, DF3

Tabella 1: *Selezioni valide per la struttura di file mostrata in Figura 2.*

(blocco chiamate in entrata e/o in uscita, ecc).
Tali condizioni per l'accesso al file non sono da considerare in modo gerarchico: in altre parole, l'aver inserito correttamente il CHV2 non permette di effettuare comandi su file che prevedono l'inserimento del CHV1.
Come molti sapranno, è possibile inserire fino a tre volte un CHV1 errato dopo le quali il codice viene bloccato. Per poter essere sbloccato, sarà necessario utilizzare un altro codice segreto denominato UNBLOCK CHV1 (PUK nella vecchia terminologia). Lo stesso vale anche per il CHV2 che ha un suo relativo UNBLOCK CHV2 (PUK2). È possibile inserire in modo errato l'UNBLOCK CHV1 o 2 un massimo di 10 volte consecutive, dopo le quali il codice bloccato non può più essere sbloccato dal cliente e la SIM deve essere

Livello	Modalità d'accesso
0	**ALW** (always, sempre): operazione sempre permessa
1	**CHV1**: operazione permessa solo dopo la verifica del codice segreto CHV1 (o se il codice **CHV1** è disabilitato)
2	**CHV2**: operazione permessa solo dopo la verifica del CHV2
3	Riservato per usi futuri
da 4 a 14	**ADM**: operazione permessa solo durante la fase amministrativa (riservata al gestore)
15	**NEV** (Never, mai): operazione mai permessa

Tabella 2: *Modalità d'accesso di un file*

riportata al gestore.
Il CHV1 può essere anche disabilitato dall'utente: in questo caso, la modalità d'accesso CHV1 sarà sempre soddisfatta.
Nessuna condizione per l'accesso è associata alle directory (DF e MF).

Comandi
Così come per tutte le smartcard ISO7816-3, il dispositivo d'interfaccia (nel caso GSM è rappresentato dall'ME) colloquia con la carta mediante dei comandi trasmessi e delle risposte ricevute. Tali comandi e risposte non sono altro che una sequenza di byte trasportata mediante un preciso protocollo, per esempio il già citato T=0.
Tutte le SIM prevedono *almeno* questo protocollo, ma possono offrire altri sistemi di trasmissione più evoluti (indicati, come al solito, nell'ATR). Per i nostri esperimenti utilizzeremo sempre il protocollo standard T=0.
Usando la terminologia dello standard ISO, la sequenza di byte che forma un comando trasmesso dall'interfaccia o la risposta della SIM viene chiamata APDU (Application Protocol Data Unit). Nel primo caso si parlerà di *command APDU* (o semplicemente comando), nel secondo di response *APDU* (o semplicemente risposta).
Normalmente, ad ogni comando c'è sempre una risposta da parte della SIM.
Come già detto nella terza parte del tutorial, un comando è formato da una intestazione (header) di cinque byte ed *eventualmente* dei dati. Nel caso siano presenti dei dati, il byte **LEN** (il quinto dell'intestazione) coincide con il numero dei byte trasferiti alla SIM. Quest'ultima risponde con due byte che rappresentano la *Status Word*, cioè il risultato dell'elaborazione del comando.
Se, invece, deve avvenire un trasferimento opposto di dati (dalla SIM all'ME), il comando sarà formato dalla sola intestazione (cinque byte) dove il byte **LEN** indicherà il numero di byte che la SIM dovrà trasmettere, mentre la risposta sarà formata da **LEN** byte di dati, più i due byte della Status Word. È da notare che, in questo caso, se LEN=0, ci sarà un trasferimento di 256 byte dalla SIM all'ME.
In realtà, la casistica è più complicata per i seguenti due motivi:

- un solo comando potrebbe prevedere il trasferimento simultaneo di dati in entrambe le direzioni;
- l'ME potrebbe non sapere a priori il numero dei byte che la SIM dovrà trasmettere, quindi non può "riempire" correttamente il byte LEN del comando relativo.

Per risolvere entrambe le situazioni, è previsto un comando dedicato (GET RESPONSE) che permette all'ME di ottenere una nuova risposta dalla SIM. Per maggiore chiarezza, in figura 3 sono rappresentati tutti i cinque casi possibili di trasferimento dati tra ME e SIM.
Il primo caso è semplice e prevede l'invio di un comando dall'ME alla SIM senza alcun dato né in uscita né in entrata. Il byte **LEN** sarà quindi nullo e la risposta della SIM conterrà solo i due byte della Status Word (90 00, nel caso non ci siano errori).

Il secondo caso è relativo al solo trasferimento di dati, di lunghezza nota, dalla SIM verso l'ME. Il comando è formato solo dall'header di cinque byte dove LEN rappresenta il numero di byte da trasferire (se LEN=0, i dati trasferiti saranno 256).

L'APDU di risposta sarà formato dai LEN byte richiesti più i due byte della Status Word.

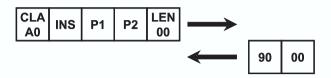
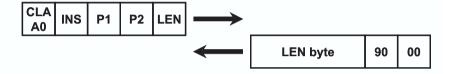
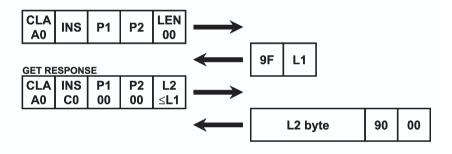
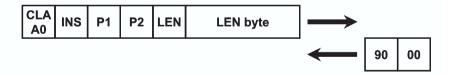
Nel terzo caso, l'ME richiede alla SIM dei dati di cui non conosce la lunghezza.

L'ME trasmette un comando con il byte **LEN**=0 e la SIM risponde con una Status Word **9F L1**, dove L1 rappresenta la lunghezza dei dati. A questo punto, l'ME può trasmettere un comando GET RESPONSE per ottenere i dati richiesti, dove specifica correttamente il byte **LEN**=L2, normalmente uguale a L1 (in alcuni casi potrebbe essere inferiore).

Anche il quarto caso è molto semplice e prevede un trasferimento di dati solo dall'ME verso la SIM.

Infine, l'ultimo caso è il più complicato, poiché prevede un trasferimento di dati dall'ME alla SIM, ma anche un trasferimento nella direzione opposta (di lunghezza conosciuta o meno). In questo caso l'ME trasmette un comando in cui invia i propri dati ed aspetta una Status Word del tipo **9F L1**, dove L1 rappresenta la lunghezza dei dati di risposta. Successivamente, l'ME invia il comando GET RESPONSE in cui riempie correttamente il byte **LEN** con L2, normalmente pari ad L1 o inferiore.

In Tabella 3 è mostrato l'elenco di tutti i comandi che una SIM riconosce, insieme alla codifica dei parametri **INS**, **P1**, **P2** e **LEN**. Il verso di trasmissione dei dati è indicato nella colonna **S/R** dove S indica che i dati sono trasmessi (Send) dall'ME

1. Nessun input/Nessun output

2. Nessun input/Output di lunghezza nota

3. Nessun input/Output di lunghezza non nota

4. Input/Nessun output

5. Input/Output di lunghezza nota o non nota

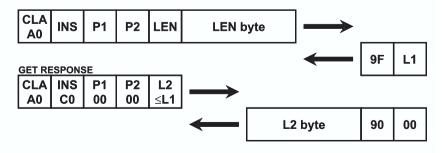



Figura 3: *Rappresentazione dei cinque casi di trasferimenti ME-SIM*

ed R indica che i dati sono ricevuti (Received) dall'ME.
Ovviamente, alcuni comandi possono prevedere entrambe le direzioni (grazie al comando GET RESPONSE).
Per l'applicazione GSM, il byte **CLA** del comando è sempre pari ad 'A0'.
Alcuni comandi gestiscono direttamente i file (EF o DF) memorizzati nella SIM, altri hanno scopi diversi (per esempio, l'esecuzione degli algoritmi A3 e A8). In Tabella 4 sono elencati tutti i comandi che riguardano i file, specificandone il tipo su cui possono essere eseguiti. Per esempio, il comando SELECT e STATUS possono essere effettuati su tutti i tipi di file, mentre il comando READ BINARY può essere eseguito solo su un transparent file.

STANCHI DI GIOCARE?

Cosa ci aspetta nel prossimo articolo

Terminiamo qui questo articolo per non renderlo troppo lungo e noioso. Nel prossimo continueremo la nostra analisi sul funzionamento delle SIM, in

COMANDO	INS	P1	P2	LEN	S/R
SELECT	A4	00	00	02	S/R
STATUS	F2	00	00	lunghezza	R
READ BINARY	B0	indirizzo alto	indirizzo basso	lunghezza	R
UPDATE BINARY	D6	indirizzo alto	indirizzo basso	lunghezza	S
READ RECORD	B2	n. record	modalità	lunghezza	R
UPDATE RECORD	DC	n. record	modalità	lunghezza	S
SEEK	A2	00	tipo/modalità	lunghezza	S/R
INCREASE	32	00	00	03	S/R
VERIFY CHV	20	00	n. CHV	08	S
CHANGE CHV	24	00	n. CHV	10	S
DISABLE CHV	26	00	01	08	S
ENABLE CHV	28	00	01	08	S
UNBLOCK CHV	2C	00	00 o 02	10	S
INVALIDATE	04	00	00	00	-
REHABILITATE	44	00	00	00	-
RUN GSM ALGORITHM	88	00	00	10	S/R
SLEEP	FA	00	00	00	-
GET RESPONSE	C0	00	00	lunghezza	R
TERMINAL PROFILE	10	00	00	lunghezza	S
ENVELOPE	C2	00	00	lunghezza	S/R
FETCH	12	00	00	lunghezza	R
TERMINAL RESPONSE	14	00	00	lunghezza	S

Tabella 3: *Elenco dei comandi di una SIM*

particolare descriveremo in dettaglio i comandi ed i file più importanti. Inoltre, tramite un apposito firmware per l'UniReader vedremo come è possibile sperimentare direttamente con le SIM senza l'utilizzo di un telefonino. Con un semplice software per PC, saremo quindi capaci di leggere e salvare sul nostro computer la rubrica e gli SMS memorizzati nella scheda.

Comando	MF	DF	Transparent	Linear-fixed	Cyclic File
SELECT	✓	✓	✓	✓	✓
STATUS	✓	✓	✓	✓	✓
READ BINARY			✓		
UPDATE BINARY			✓		
READ RECORD				✓	✓
UPDATE RECORD				✓	✓
SEEK				✓	
INCREASE					✓
INVALIDATE			✓	✓	✓
REHABILITATE			✓	✓	✓

Tabella 4: *Comandi relativi ai file.*

BIBLIOGRAGIA E LINK

Normative GSM (in inglese): www.etsi.org
Standard ISO7816 (in inglese, a pagamento): www.iso.org

PROGETTIAMO UN RAZZO
nona parte

ALTIMETRO BAROMETRICO A DOPPIA ESPULSIONE

di Esteban Mascarella *(net03593@cr-surfing.net)* **e di Eugenio Cosolo** *(info@missilistica.it)*
(si ringraziano per la collaborazione Stefano Innocenti e Alessio Cosolo)

In questa nuova puntata della rubrica di missilistica amatoriale illustreremo un progetto tecnologicamente molto avanzato, che ha richiesto diversi mesi di lavoro per la scrittura del software in quanto impiega algoritmi molto sofisticati. Si tratta di un altimetro elettronico computerizzato in grado di rilevare la quota massima raggiunta da un razzo e di attivare due distinti sistemi di espulsione dei paracadute, uno all'apogeo ed uno alla quota di 200 metri. Il valore della quota massima viene memorizzato per essere poi rivelato con una serie di beep codificati. È un dispositivo indispensabile per i razzi più evoluti e per quelli non dotati di un sistema di espulsione autonomo, come quelli propulsi da un motore ibrido. Le dimensioni molto contenute lo rendono comunque adatto per qualsiasi modello di razzo, anche quelli più piccoli. La semplicità costruttiva ed il costo modesto permettono la realizzazione a chiuque.

PRINCIPIO DI FUNZIONAMENTO

Come sappiamo, la pressione atmosferica varia in funzione dell'altitudine, perciò possiamo sfruttare questo fenomeno fisico per misurare la quota a cui siamo, misurando con precisione la pressione ambientale.

Purtroppo le cose sono leggermente più complesse, in quanto per motivi meteorologici, la pressione teorica ricavata dalla tabella dell'ATMOSFERA STANDARD varia notevolmente in funzione delle condizioni meteorologiche, come il susseguirsi delle alte e basse pressioni.

È perciò indispensabile prima tarare l'altimetro ad una quota conosciuta e poi effettuare la lettura a quote diverse. La taratura è ovviamente destinata a deteriorarsi in poco tempo ma per le nostre esigenze è sufficiente conoscere la pressione iniziale del punto di lancio.

Il processore svolge automaticamente questa funzione: al momento dell'accensione campiona la pressione ambientale e la memorizza come "quota zero".

Se ad esempio siamo a quota 30 metri sul livello del mare e la pressione rilevata è di 1000 millibar, questo valore sarà considerato come base (quota zero) per i successivi campionamenti per stabilire la quota differenziale. Dopo l'azzeramento il processore verifica le letture successive per stabilire se effettivamente è iniziato il volo. Se la pressione diminuisce per l'equivalente di 100 metri di quota, decide che il volo è iniziato e "arma" il dispositivo di espulsione del paracadute. Questa fase serve a introdurre un fattore di sicurezza in modo da non rischiare di espellere il paracadute per errore. Lievi variazioni di pressione possono essere causate ad esempio da refoli di vento, che potrebbero trarre in inganno il sistema facendogli credere di essere già in volo.
A questo punto il microprocessore attende che il decremento di pressione termini, il che significa che la quota massima è stata raggiunta. In questo istante attiva il canale collegato all'espulsione del paracadute "drogue" e poi continua a rilevare la pressione ambientale, questa volta in aumento, visto che siamo in fase di discesa.
Quando secondo i suoi calcoli la quota raggiunge i 300 metri attiva la seconda uscita, quella collegata all'espulsione del paracadute principale che ha il compito di rallentare la discesa ad una velocità di sicurezza. Al termine del volo il processore inizia una sequenza acustica in codice per informarci sulla quota massima raggiunta (vedi i dettagli in seguito).

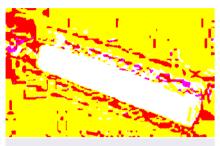



Figura 1: Foto del paracadute

LA PRESSIONE ATMOSFERICA IN FUNZIONE DELLA QUOTA

Abbiamo visto che un altimetro

barometrico stabilisce la quota in funzione della pressione atmosferica.
Possiamo ricavare la pressione teorica (valida fino a circa 6000 metri) dalla seguente formula:
Pq = P0 * (EXP ((H / 8000 * - 1)))

Dove:

- **Pq** = pressione atmosferica in quota (mBar)
- **P0** = pressione atmosferica alla quota di lancio (mBar)
- **H** = quota (metri)

La formula che ci permette di risalire alla differenza di pressione in base al codice audio emesso dal microprocessore è la seguente:

delta P = codice altimetro * 1,0845

Il valore di 1,0845 è ricavato dalle caratteristiche del sensore di pressione e dalla conversione AD del microprocessore.
Se ad esempio l'altimetro emette il codice 75, possiamo calcolare che la variazione della pressione atmosferica tra la quota zero e l'apogeo è di:

delta P = (75 * 1,0845) =
= 81,34 mBar

Considerata P0 la pressione della quota di lancio (supponiamo 1013,25 mBar), la pressione assoluta all'apogeo è di:

Pq = P0 – delta P = 1013,25 – 81,34 = 931,91 mBar

(La leggera differenza di valori rispetto alla tabella del ATMOSFERA STANDARD è dovuta agli arrotondamenti introdotti nei calcoli).

La pressione ottenuta può essere convertita in metri di quota applicando la seguente formula:
H = 8000 * (log e (P0 / Pq))

Per comodità abbiamo allestito un semplice foglio Excel per il calcolo immediato di tutti i parametri, con step di 25 metri, da scaricare dal nosto sito Web: www.farelettronica.com
La tabella può essere stampata e usata "sul campo" per ottenere il valore effettivo della quota raggiunta.

SISTEMI DI ESPULSIONE

Esistono diversi metodi per espellere il paracadute dal razzo, il più semplice, usato soprattutto sui modelli di piccole dimensioni, consiste nel far esplodere una micro carica di polvere nera che serve a pressurizzare il vano contenente il paracadute ed eiettarlo all'esterno previa separazione dell'ogiva. La quantità di polvere nera generalmente usata è di un grammo o anche meno, a seconda del volume interno da pressurizzare.
La reperibilità della polvere (usata per le armi ad avancarica da collezione, che negli USA chiamano FFFF o 4F) è piuttosto difficoltosa in quanto per acquistarla presso le armerie è necessario essere in possesso del porto d'armi. Questa polvere sviluppa una enorme quantità di gas, ed è abbastanza "lenta"; per questi motivi viene usata per la pressurizzazione dei volumi.
L'altra soluzione sarebbe quella di prepararsela da soli ma per ovvi motivi non è possibile indicare in questa sede le modalità di confezionamento ed in ogni

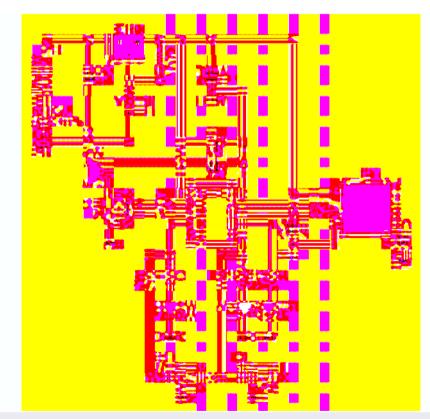



Figura 2: Schema elettrico

Elenco componenti			
Sigla	**Valore**	**Sigla**	**Valore**
R1, R4, R6	10 KΩ 1/4 W	T3	BC337
R2, R8	270 Ω 1/4 W	Bz1	Buzzer piezoelettrico 5 V
R3	1 KΩ trimmer multigiri	LD1	LED verde 3 mm
R5, R7, R9	470 Ω	LD2	LED rosso 3 mm
C1, C2, C5, C6	100 nF poliestere	LD3	LED giallo 3 mm
C3	10 µF elettrolitico	U1	PIC12F675
C4	1000 pF ceramico	U2	Sensore barometrico Fujikura XFPM-115kpa
T1, T2	Mosfet BUK8524-55	Reg1	LM317L

caso la procedura richiederebbe dei requisiti di sicurezza non in possesso dello sperimentatore amatoriale.

Possiamo invece accennare al funzionamento degli accenditori elettrici (e-match) che vengono usati per attivare la carica. Si tratta di dispositivi composti da un sottile filamento di nikelcromo che sottoposto ad una corrente elettrica diventa incandescente. Il calore prodotto è sufficiente ad innescare la combustione della polvere nera o dell'apposito materiale pirogeno che circonda il filamento.

Un metodo molto semplice è quello di fissare sui terminali di una coppia di fili isolati in plastica (va benissimo il filo citofonico) uno spezzone lungo qualche millimetro di capillare in nikelcromo, recuperato ad esempio da una resistenza elettrica di bassa potenza.

Una soluzione alternativa è quella di rompere la punta di una lampadina per alberi di natale funzionante a bassa tensione (3 volt) ed introdurla nella polvere nera, raccolta in un sacchetto di carta.

Per i razzi di maggiori dimensioni vengono usati dei dispositivi di espulsione basati su principi diversi, come ad esempio l'apertura di portelli e l'espulsione meccanica per mezzo di molle.

In questo caso l'impulso di comando servirà ad attivare dei servomotori o dei meccanismi elettromagnetici.

In qualsiasi caso, la sorgente di energia per l'accensione o attivazione dovrà essere fornita esternamente al circuito dell'altimetro, in quanto i mosfet agiscono da semplici interruttori e la microbatteria di alimentazione non è in grado di erogare forti correnti. È perciò necessario prelevare l'ali-

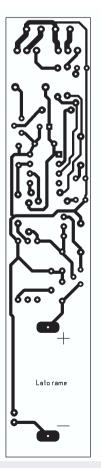



Figura 3: Circuito stampato scala 1:1 (lato rame)

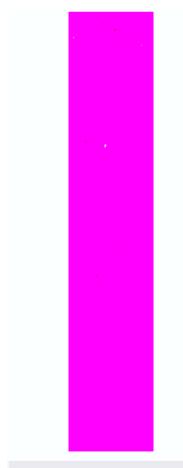



Figura 4: Piano di montaggio dei componenti

mentazione da una batteria da 9 volt, alcalina.

PRINCIPIO DI FUNZIONAMENTO

Il cuore del sistema è un microcontroller PIC12F675. Pur nelle dimensioni ridotte (ha solo 8 piedini) questo chip dispone di 6 pin programmabili I/O (entrata / uscita), un convertitore analogico a 10 bit, un oscillatore interno, timer programmabili, 1K words di memoria a tecnologia Flash (riprogrammabile), 64 Bytes di memoria RAM, 128 Bytes EEPROM e altre periferiche minori.
La pressione atmosferica viene misurata da un sensore di pressione *XFPM-115KPA Fujikura* che eroga in uscita una tensione proporzionale alla pressione ambientale, l'unico componente esterno necessario a questo trasduttore è un condensatore.
L'uscita analogica di questo sensore è collegata al pin 7 (GP0, programmato come entrata analogica) del PIC attraverso un filtro composto da R1 e C1.
I pin 2, 3, 4, 5, 6 (GP5, GP4, GP3, GP2 e GP1) sono invece programmati come uscite.
Il programma effettua una lettura iniziale dopo 5 secondi dall'accensione per determinare la quota zero, considerata come quota di riferimento.
Raggiunta la quota minima prestabilita di 100 metri, il sistema si "arma" e si accende il LED verde (armed), questo serve al sistema per sapere che il razzo è decollato.
Poi il microcontroller determina se il razzo ha raggiunto l'apogeo (la quota massima dell'inviluppo di volo) e per far questo il PIC continua a calcolare la quota attuale e la compara con la lettura precedentemente memorizzata, in modo da capire se il razzo è ancora in fase di salita.
Se per un certo numero di volte la lettura è inferiore all'ultimo valore memorizzato significa che il razzo ha raggiunto l'apogeo e perciò è iniziata la discesa.
A questo punto viene attivato per un secondo il pin 3, sul quale è collegato ad un mosfet di potenza che ha lo scopo di attivare la carica di espulsione per aprire il paracadute secondario (Drogue).
Poi il programma continua a leggere la quota e quando viene raggiunta un'altra quota predeterminata, minore a quella di apogeo, viene attivato il pin 5 per un secondo, anche questo collegato un mosfet di potenza che funge da interruttore per attivare l'espulsione del paracadute principale (Main).
Infine il programma comincia a riferire il valore della quota massima per mezzo di una serie di beep audio emessi dal buzzer.
La codifica usata è la seguente: 1 beep per la cifra uno, 2 beep per il due, ecc. 10 per lo Zero.
Dopo ogni cifra è inserita una pausa, mentre una pausa più lunga indica l'inizio della sequenza.
Il gruppo di quattro cifre (il valore massimo della lettura è 1023) rappresenta il valore memorizzato nel microprocessore, ma deve essere convertito in metri di quota utilizzando la tabella 2.
La lettura viene ottenuta sottraendo dalla quota rilevata la quota zero, pertanto la lettura è relativa alla quota di lancio.
L'alimentazione viene fornita da una minibatteria a 12 V oppure da una pila rettangolare da 9 V, regolata a 5 V dallo stabilizzatore LM317L. Il trimmer multigiri R3 da 1K serve a regolare questo valore con precisione.
Il motivo per cui è stato usato un LM317 al posto del classico

Numero beep	Cifra equivalente
10	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Ad esempio, la sequenza acustica: beep-beep-beep-beep-beep-beep-beep-beep-beep-beep-PAUSA-beep-PAUSA-beep-beep-beep-beep-PAUSA-beep-beep, fornisce il codice **0142**, che in base alla tabella di conversione corrisponde alla quota di **1325** metri.

Tabella 1: *Codice sonoro*

78L05 è la maggiore deriva termica di quest'ultimo e poichè la tensione di 5 V serve anche come riferimento per il convertitore analogico interno al PIC. Se questo non è preciso anche la conversione perderà in precisione.

LA REALIZZAZIONE PRATICA

L'assemblaggio del circuito è facilitato dal circuito stampato illustrato in figura 3 e 4, da realizzare con le solite tecniche.
Le dimensioni sono davvero ridotte in modo da poter installare l'altimetro anche su modelli di piccole dimensioni.
Le saldature dovranno essere molto accurate poichè il circuito sarà sottoposto a violente accelerazioni in fase di decollo e atterraggio e questo potrebbe causare dei guasti nel caso queste fossero precarie.
Il sensore barometrico non è proprio di facile reperibilità ma una ricerca su internet vi permetterà di rintracciare diversi distributori della Fujikura.
È importante che l'interno del vano nel quale viene installato l'altimetro non sia ermetico, in quanto la pressione interna deve equivalere a quella esterna.
Per ottenere questo è necessario praticare dei fori sul perimetro del vano payload, in una zona non disturbata da turbolenze, come potrebbe essere quella prossima all'ogiva.
La cosa migliore sono tre fori da 4 o 5 mm di diametro angolati di 120 gradi, ad almeno 100 mm dall'inizio della sezione cilindrica del razzo.

CONCLUSIONI

Il codice per programmare il microprocessore, come il foglio di calcolo per la conversione sonoro/metri, possono essere liberamente scaricati dal sito di Fare Elettronica.
In caso di difficoltà, è anche possibile richiedere il microprocessore già programmato direttamente all'autore.

Codice	Metri	Codice	Metri
3	25	59	525
6	50	62	550
8	75	65	575
11	100	67	600
14	125	70	625
17	150	73	650
20	175	75	675
23	200	78	700
26	225	81	725
28	250	83	750
31	275	86	775
34	300	89	800
37	325	91	825
40	350	94	850
43	375	97	875
45	400	99	900
48	425	102	925
51	450	104	950
54	475	107	975
56	500	109	1000

Tabella 2: *Tabella semplificata di conversione codice sonoro/metri di quota (la tabella completa, in formato Excel, è possibile scaricarla dal sito Web di Fare elettronica)*

Le fiere e mostre mercato di Novembre

2004

06-07 Novembre 2004
FIERA ABC DELL'ELETTRONICA

La Fiera A.B.C. dell'Elettronica si svolge due volte all'anno, in primavera ed in autunno. Oltre alle merceologie "tradizionali" proposte da questo tipo di manifestazioni, quali computer, elettronica in genere, radiantismo, telefonia, surplus... nonché radio d'epoca, dischi e CD da collezione.
Inoltre l'A.B.C. dell'Elettronica propone, in primavera, il Salone dell'Astronomia e Photo Cine video, con macchine ed attrezzature per la fotografia , mentre in autunno i protagonisti sono i radioamatori grazie al CB Day, a loro dedicato.
Certamente non mancano i buoni motivi per andare a dare un'occhiata; ci saranno buoni affari sia per gli esperti sia per i neofiti! La Fiera A.B.C. dell'Elettronica appartiene al circuito Expo Elettronica.

Luogo: Lario Fiere – Viale Resegone – Erba (CO)
Orari: 9.00/18.00
Organizzatore: Blu Nautilus (www.blunautilus.it Tel. 0541-53294)
Ingresso: Intero € 7,00 - ridotto € 6,00

06-07 Novembre 2004
ROMA HI-END 2004

Il ROMA HI-END 2004 è una manifestazione dedicata agli appassionati del "solo audio Hi-Fi" (non home-theatre), ai professionisti del settore, ai venditori, ed è promossa dalle case produttrici di alta fedeltà e dalle riviste del settore. L'obbiettivo è quello di offrire ai visitatori la possibilità di venire in contatto e di confrontarsi in diretta con gli operatori del settore, ascoltare le apparecchiature hi-end, fare esperienza delle tecnologie dell'audio di alta qualità, conoscere e ascoltare la musica "alla sorgente".

Luogo: Centro Congressi Midas Jolly Hotel
Via Aurelia, 800 - ROMA
Orari: 9.00/18.00
Organizzatore: The Sound Of The Valve (www.thesoundofthevalve.it)
Ingresso: Gratuito

13-14 Novembre 2004

ELETTROEXPO

33^ Mostra mercato di elettronica, radiantismo, strumentazione, componentistica informatica. Settori merceologici: Materiale radiantistico per CB e radiomatori, apparecchiature per telecomunicazioni, telefonia, antenne e parabole per radiomatori e tv Sat, radio d'epoca, stampa specializzata.

LUOGO: Quartiere Fieristico - Verona
ORARI: 9.00/18.00
ORGANIZZATORE: Verona Fiere (www.veronafiere.it Tel. 045-8298111)
INGRESSO: n.p.

21 Novembre 2004

RADIOAMATORE 2

Mostra mercato di radiantistica, elettronica, componentistica, impianti radio e di teletrasmissione. Spazio all'home computer e a tutte le applicazioni dell'informatica ad uso personale: hardware, software, videogiochi, internet. I numeri di Radioamatore 2 dello scorso anno: 9.000 mq coperti di superficie espositiva, 139 espositori diretti ed oltre 16.943 visitatori.

LUOGO: Quartiere Fieristico – Pordenone
ORARI: 9.00/18.00
ORGANIZZATORE: Pordenone Fiere (www.fierapordenone.it Tel. 0434-232111)
INGRESSO: n.p.

27-28 Novembre 2004

XXXIX MOSTRA MERCATO NAZIONALE DEL RADIOAMATORE

LUOGO: Fiera Adriatica – Silvi Marina (TE)
ORARI: 9.00/19.00
ORGANIZZATORE: A.R.I. Pescara (www.aripescara.it Tel. 085-4714835)
INGRESSO: n.p.

4-5 Dicembre 2004

GRANDE FIERA DELL'ELETTRONICA

La Grande Fiera dell'Elettronica di Forlì, con due edizioni annuali, è fra le manifestazioni più famose e frequentate a livello nazionale. L'edizione di primavera propone, oltre all'elettronica, il Flight simulator show: gare e dimostrazioni di volo simulato per provare l'ebbrezza di essere alla guida di un veivolo... con i piedi ben a terra! Per gli appassionati di collezionismo radio d'epoca, dischi e CD da collezione. L'appuntamento di dicembre invece è una vera e propria kermesse con oltre 350 aziende partecipanti: elettronica, fotografia tradizionale e digitale, radio d'epoca, dischi e cd da collezione, il concorso nazionale dell'inventore elettrico ed elettronico e il Salone dell'Astronomia che in pochi anni ha conquistato il consenso di espositori e pubblico.

LUOGO: Fiera di Forlì – Via Punta di Ferro – Forlì (FC)
ORARI: 9.00/18.00
ORGANIZZATORE: Blu Nautilus (www.blunautilus.it Tel. 0541-53294)
INGRESSO: Intero € 7,00 - ridotto € 6,00

PRATICAMENTE... OSCILLATORI AL QUARZO: CONTASECONDI

di Maurizio Del Corso
m.delcorso@farelettronica.com

Come può un cristallo di quarzo oscillare ad una determinata frequenza? Quale spiegazione fisica c'è dietro a questo fenomeno? Perché si usano i quarzi per la realizzazione di onde quadre? Queste sono solo alcune delle domande che troveranno risposta nella puntata di questo mese. Cercando di capire come funzionano questi dispositivi, realizzeremo un semplice "contasecondi".

ANALISI E SPECIFICHE DEL PROBLEMA

Si vuol realizzare un circuito in grado di scandire i secondi e fornire il conteggio in forma binaria su 4 bit. I secondi dovranno essere scanditi con una precisione piuttosto elevata. Il dispositivo dovrà consentire la modularità ovvero la possibilità di espandere il conteggio oltre i 10 secondi.

LA SOLUZIONE PROPOSTA

La soluzione proposta è riportata in figura1 e si basa sull'utilizzo di un quarzo per garantire una grande precisione nel conteggio dei secondi. Il circuito integrato IC1 è un divisore di frequenza a 14 bit, ciò significa che la frequenza del segnale di ingresso può essere divisa per un fattore massimo di 2^{14}=16384. Inserendo in ingresso un segnale a frequenza di 32,768KHz si ottiene in uscita un segnale di frequenza pari a 2Hz (dato appunto da

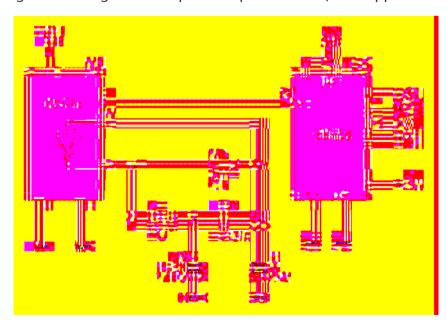



Figura 1: La soluzione proposta

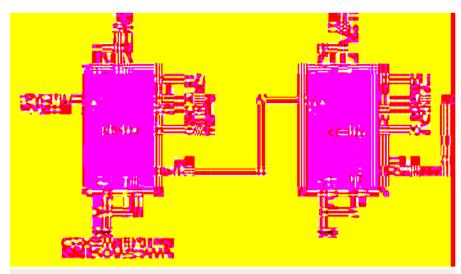



Figura 2: Espansione del numero di cifre

32768/16384). Il segnale così ottenuto viene inviato all'ingresso di clock di IC2 che è un contatore decimale il quale scandisce il conteggio dei secondi.
Il pin 16 di IC2 (U/D) determina il verso del conteggio: se collegato a livello alto il conteggio è in avanti, viceversa il conteggio viene effettuato all'indietro. Il pin 9 (B/D) permette di selezionare la modalità di conteggio: se a livello alto il conteggio è binario (quindi da 0 a 15), mentre se a livello basso è decimale (da 0 a 9). Q1÷Q4 sono le uscite, di cui Q4 è la più significativa. Cout è a livello alto durante il conteggio e commuta a zero alla fine del conteggio (15 se la modalità è binaria, 9 se decimale). La modularità si ottiene collegando il pin Cout di IC2 all'ingresso di clock di un secondo contatore decimale, come mostrato in figura 2. In questo modo si può espandere il conteggio fino al numero di cifre volute.
Se avete seguito la rubrica "Praticamente" fin dalla prima puntata, non avrete difficoltà a collegare, alle uscite dei contatori, un driver per display a 7 segmenti in modo da visualizzare le varie cifre del conteggio (vedi "Praticamente" FE 224 - Febbraio 2004).

Figura 3: Struttura di un quarzo per applicazioni elettroniche.

E IL QUARZO?

Eccoci al punto… il quarzo. Il quarzo è un materiale piezoelettrico ovvero ha la caratteristica di generare una piccola tensione se sollecitato meccanicamente e, viceversa, deformarsi se sottoposto ad una tensione esterna. Fisicamente un quarzo è costituito da una lamina di materiale piezoelettrico (quarzo, appunto) ai cui estremi sono stati ricavati due elettrodi in modo da poter applicare una

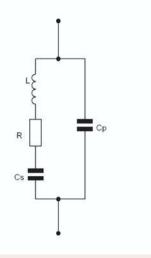



Figura 4: Circuito equivalente di un quarzo.

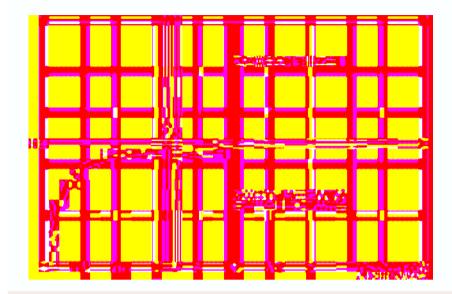



Figura 5: Impedenza di quarzo da 32,768 KHz in funzione della frequenza.

tensione dall'esterno (figura 3). Elettricamente il quarzo può essere considerato come un circuito risonante RLC come mostrato in figura 4. Poiché nel circuito equivalente del quarzo vi sono induttanze e condensatori, è evidente che la sua impedenza dipende dalla frequenza dei segnali che lo attraversano. Tracciando un grafico dell'impedenza (più correttamente della reattanza) al variare della frequenza si ottiene un andamento come quello riportato nella figura 5.
Il grafico di figura 5 deve essere interpretato nel seguente modo: nell'intervallo di frequenze in cui il grafico sta al di sopra dell'asse orizzontale (quindi tra F_1 ed F_2), il circuito di figura 4 ha un comportamento prevalentemente induttivo (si comporta come una induttanza), mentre per gli altri intervalli di frequenza il circuito ha un comportamento capacitivo (si comporta come un condensatore). È importante notare che l'intervallo di frequenze in cui il comportamento è di tipo induttivo, è molto ristretto.

OSCILLATORE DI PIERCE

Un circuito in grado di generare un'onda quadra impiegando un quarzo, è detto oscillatore di Pierce ed ha la struttura riportata in figura 6. Tralasciando la teoria che sta dietro ai circuiti oscillanti (elaborata da un certo signor Barkhausen), il circuito di figura 6 può oscillare solo se tra i due condensatori si trova un'induttanza. Poiché il quarzo si comporta come induttanza solo in un range di frequenza molto stretto, la frequenza dell'onda quadra di uscita sarà senz'altro compresa tra i valori F_1 ed F_2.
La precisione di un oscillatore di questo tipo è dell'ordine di una parte su un milione (il che significa commettere un errore di 1Hz su una frequenza di 1MHz). La porta NOT unita alla resistenza Rf, forma un amplificatore ad alto guadagno, mentre la resistenza R limita la corrente di uscita della porta. Sostituendo uno dei due condensatori con un condensatore variabile, è possibile agire, anche se in minima parte, sul valore della frequenza di uscita.
Esaminando attentamente il circuito di figura 1 è possibile riconoscere quello di figura 6: la porta NOT è infatti interna ad IC1, R2 svolge la funzione di Rf, R1 limita la corrente di uscita della porta (funzione svolta da R in figura 6) e C1, C2 e XTAL sono i due condensatori (di cui uno variabile) ed il quarzo che costituiscono il cuore dell'oscillatore. Normalmente per C1 e C2 si scelgono valori di qualche decina di picoFarad. Se avete avuto a che fare con i microcontrollori PIC, avrete senz'altro riconosciuto il circuito formato dai due condensatori ed il quarzo usati per il clock del micro.
Un generatore di clock realizzato con un quarzo ha alcuni vantaggi rispetto ad uno realizzato con un multivibratore astabile ad operazionale o con timer

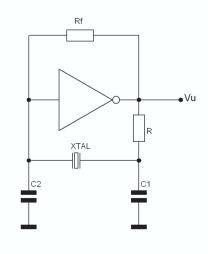



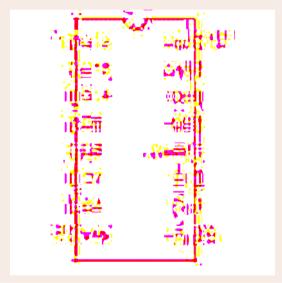
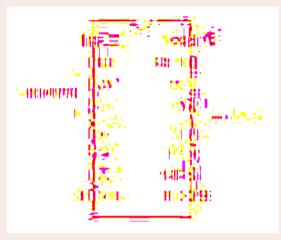
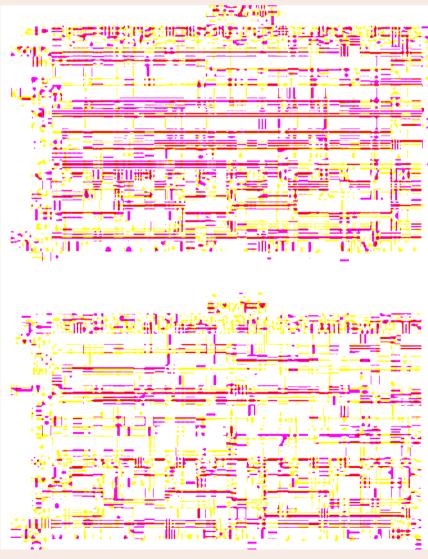
Figura 6: Oscillatore di Pierce

555. Il primo è la precisione e la stabilità della frequenza del segnale ottenuto che, come già detto, è dell'ordine di una parte su un milione. Altro vantaggio è la possibilità di cambiare la frequenza di oscillazione semplicemente sostituendo il quarzo con uno di diversa frequenza caratteristica oltre al fatto che con un oscillatore quarzato si possono ottenere frequenze molto più elevate rispetto a quelle ottenute con un astabile.

IL QUESITO DEL MESE

Risolviamo prima il quesito del numero 231. La presenza dei due diodi D1 e D2 fa sì che nel processo di carica e scarica del condensatore C siano coinvolti diverse resistenze. Durante la fase di carica la corrente fluisce da Vcc verso il condensatore per cui D2 sarà in conduzione e D1 interdetto quindi saranno coinvolte la resistenza Ra (connessa a +Vcc) e la resistenza Rb (in serie a D2). Nella fase di scarica la corrente è uscente dal condensatore (scorre verso l'alto) quindi D1 è in conduzione e D2 interdetto. In questo caso le resistenze coinvolte sono ancora Ra ed Rb, ma questa volta sono quelle in serie a D1. Questa modifica permette quindi di ottenere un'onda quadra con duty cycle del 50%.

Come quesito del mese propongo di modificare il circuito di figura 1 in modo da accendere per un istante un LED ogni 10 secondi.

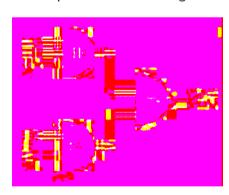
CD4029BC - PRESETTABLE BINARY/DECADE UP/DOWN COUNTER

CD4060B - CMOS 14-STAGES RIPPLE CARRY BYNARY COUNTER/DIVIDER AND OSCILLATOR

DI VIDEOSORVEGLIANZA

WIRELESS

GE.CO.AS. GENERATORE DI CODICE ASSEMBLER PER MICROCONTROLLORI PIC

di Dario Mazzeo
dmazzeo@ingele.com

Ge.Co.As. è un software pensato per soddisfare le esigenze degli hobbysti nella programmazione dei microcontrollori. Grazie alla sua interfaccia semplice da usare, chiunque potrà programmare un microcontrollore in poco tempo e senza alcuna spesa.

Il software illustrato in questo articolo, consente di programmare un microcontrollore PIC senza la necessità di conoscere un linguaggio di programmazione o competenze di base.
Prima di procedere alla descrizione del software, sarà utile definire il concetto di *Tabella di verità*. Dati degli ingressi, e fissate delle uscite, si definisce Tabella di verità una tabella che mette in relazione ad ogni combinazione degli ingressi un determinato valore delle uscite. Pertanto, il numero di righe che costituiranno la tabella, sarà rappresentato da 2n mintermini della funzione (righe della tabella), dove *n* è il numero degli ingressi.
A titolo di esempio, si immagini di voler realizzare una rete logica come quella illustrata di seguito.

Come si può notare, il circuito logico è caratterizzato dagli ingressi A÷E e dall'uscita OUT. Per risolvere tale rete, occorreranno due circuiti integrati AND e OR connessi opportunamente.
Com'è noto, ogni rete logica deve essere resa in forma minima mediante opportuni metodi di minimizzazione. Per ridurre tali inconvenienti, è possibile utilizzare il software Ge.Co.As. e realizzare in pochi minuti un integrato che realizzi tale funzione.
Nella figura 1, l'uscita (OUT) della funzione assume valore logico alto quando gli ingressi ABCD o ABCE sono affermati e valore logico basso in tutti gli altri casi.

ESEMPIO DI RETE LOGICA

Prima dell'avvio del software, sarà necessario associare gli ingressi e le uscite della rete in esame, agli ingressi e le uscite fisiche del microcontrollore. In questo contesto, verrà scelto RA0, RA1, RA2, RA3, RA4, per gli ingressi A, B, C, D, E e RB0 per l'uscita OUT.
Avviando il software, basterà selezionare gli ingressi e le uscite per i mintermini ABCD e ABCE, come

mostrato in figura 2, e premere il pulsante “Aggiungi”.

Sulla parte destra dell’interfaccia è possibile notare la Tabella di verità con le 32 righe (2^5 dei relativi 5 ingressi) e nella parte inferiore il codice generato che realizza la funzione sopra descritta.
A questo punto sarà sufficiente premere il bottone *“Genera codice assembler”* e *“Compila codice assembler”*, per generare il firmware e compilarlo in formato HEX per la programmazione.

ESEMPIO DI DECODER

In questo esempio ci prefiggiamo di personalizzare le funzionalità di decodifica offerte dall’integrato 4511, mediante l’uso del microcontrollore.
Volendo realizzare un decoder che fornisca una rappresentazione decimale mediante un ingresso binario a 5 bit, sarà possibile definire la tabella di verità di tutti i mintermini e delle relative uscite, tenendo presente i segmenti del display da illuminare e il valore decimale corrispondente agli ingressi binari.
Prima di procedere con l’inserimento dei valori, sarà necessario che le uscite del microcontrollore siano collegate ai relativi segmenti del display LCD (a catodo comune) e che gli ingressi RA0÷RA4 siano predisposti in modo tale che RA0 abbia un peso pari a 2^0, RA1 a 2^1, RA2 a 2^2 e così via.
L’interfaccia presenta dei tools per agevolare tali operazioni di progettazione e definizione della tabella di verità, come mostrato in figura 4. È possibile definire il valore decimale assunto dagli

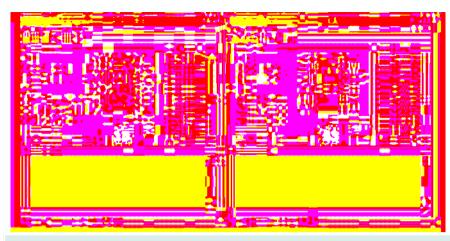



Figura 2: *Selezione degli ingressi e delle uscite*

ingressi, mediante la formula:

$N = RA0 \times 2^0 + RA1 \times 2^1 + RA2 \times 2^2 + RA3 \times 2^3 + RA4 \times 2^4$

dove gli ingressi RA0÷RA4 potranno assumere valore 1 o 0 se presenteranno all’ingresso rispettivamente un valore logico affermato o negato.

Bisogna specificare che i livelli di tensione per gli ingressi RA0:RA4, dovranno essere posti ad un valore logico alto tramite una rete di pull-up. In questo caso, ad un valore logico affermato, rappresentato sull’interfaccia del software, corrisponderà un livello di tensione basso. Viceversa, in caso di rete di pull-down, sarà sufficiente

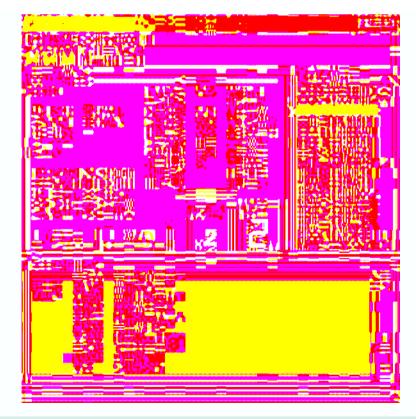



Figura 3: *GE.CO.AS. Schermata principale*

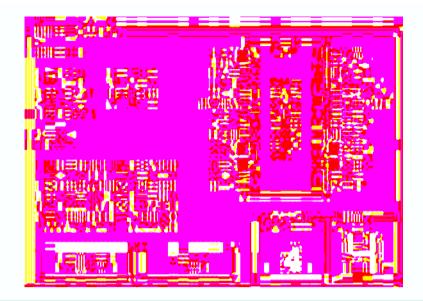



Figura 4: *Configurazione degli ingressi e delle uscite*

disabilitare l'opzione *"Complementa ingressi"* in *"Compilazione"*.

Collegando le uscite RB0÷RB7 ai rispettivi segmenti A÷H del display LCD, sarà possibile visualizzare sull'interfaccia del software un'anteprima di come questi si illumineranno.

Si selezionerà quindi: RA0, RB1 e RB2 per abilitare i segmenti relativi al valore decimale 1; RA1, RB0, RB1, RB3, RB4, RB6 per abilitare i segmenti relativi al valore decimale 2, e così via, fino al completamento della Tabella di verità.

AGGIUNTA DI RIGHE

Per aggiungere una o più righe nella Tabella di verità, sarà necessario specificare il valore delle uscite in corrispondenza dei valori degli ingressi e premere sul tasto "Aggiungi".
Abilitando l'opzione "Azzera campi dopo inserimento", sotto la voce di menù "Modifica", è possibile azzerare tutti i valori ad ogni inserimento.
Esempio, per abilitare le uscite RB0, RB1 e RB2 in corrispondenza del valore logico affermato RA0, bisognerà:

- Abilitare l'ingresso RA0.
- Abilitare gli ingressi RB0, RB1 e RB2.
- Premere il tasto "Aggiungi".

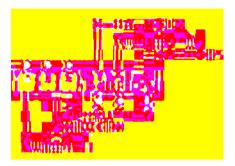
Successivamente la riga verrà inserita opportunamente nella Tabella di verità.

MODIFICA DI RIGHE

Per modificare una riga presente nella Tabella di verità, è necessario:

- Cliccare due volte su di essa.
- Applicare le modifiche volute.
- Cliccare sul tasto "Aggiungi".

CANCELLAZIONE DI RIGHE

Per cancellare la Tabella di verità cliccare sulla voce di menù: "Modifica" e "Cancella Tabella", oppure sulla relativa icona come mostrato in figura 5.

SALVATAGGIO DELLA TABELLA DI VERITÀ

È possibile salvare la Tabella di verità cliccando sulla voce di menù "File" e "Salva tabella...", oppure sulla relativa icona (figura 5).
In tutti e due i casi sarà necessario specificare il percorso e il nome del file, seguito dall'estensione ".gca" su cui salvare il risultato.

SALVATAGGIO DEL CODICE ASSEMBLER

È possibile salvare il risultato della generazione del codice Assembler, cliccando sulla voce di menù "File" e "Salva codice...", oppure sulla relativa icona (figura 5).
In tutti e due i casi sarà necessario specificare il percorso e il nome del file seguito dall'estensione ".asm" su cui salvare il risultato.

RIPRISTINO DEL LAVORO SALVATO

È possibile recuperare il lavoro salvato cliccando sulla voce di menù "File" e "Apri", oppure cliccando sulla relativa icona (figura 5).
In tutti e due i casi sarà necessario specificare il percorso e il nome del file (.gca) precedentemente salvato. L'utente non verrà avvisato della perdita del lavoro corrente.

GENERAZIONE CODICE ASSEMBLER

Dalla Tabella di verità è possibile

generare il codice Assembler per il microcontrollore, cliccando sulla voce di menù "Compilazione" e "Genera codice Assembler", oppure sulla relativa icona (figura 5).
Il risultato della compilazione verrà visualizzato nel campo di testo in fondo alla finestra.

COMPILAZIONE CODICE ASSEMBLER

La compilazione del codice Assembler avverrà mediante il compilatore "MPASM.exe" presente nella directory "C:\Programmi\MPLAB\" e scaricabile gratuitamente dal sito internet www.microchip.com.
Per avviare la compilazione, sarà sufficiente cliccare su "Compila codice Assembler" nella voce di menù "Compilazione", oppure sulla relativa icona (figura 5).
In tutti e due i casi bisognerà specificare il percorso e il nome del file Assembler (.asm) su cui salvare il risultato della compilazione.

OPZIONI DI COMPILAZIONE

L'opzione "Complementa Ingressi" nella voce di menù "Compilazione" deve essere abilitata qualora il microcontrollore lavorasse in logica negata, ovvero, se i terminali di ingresso sono posti al valore logico affermato (+5V) tramite una rete di pull-up.
Per default l'opzione "Complementa ingressi" è abilitata.

INSTALLAZIONE

Installare il software Ge.Co.As. e il software della MicroChip MPLAB nelle directory predefinite.
Ad installazione ultimata controllare che la guida in linea e la compilazione del software Ge.Co.As. funzionino correttamente. In caso contrario procedere in questo modo:

- Con il tasto destro del mouse sull'icona di Ge.Co.As., cliccare su Proprietà ed inserire nel campo "Da:" lo stesso percorso contenuto nel campo "Destinazione:" senza il nome del file. Ovvero, per la directory predefinita il testo da inserire è: "C:\Programmi\GeCoAs\".
- Nella directory predefinita di Ge.Co.As., aprire il file "path.txt" e controllare che il percorso del software MPLAB, contenuto nel file, coincida con quello presente sul computer.

CONCLUSIONI

Il software mostrato in questo articolo è in grado di generare il codice assembler per i microcontrollori PIC della serie 16x84 ed estendersi a qualsiasi serie modificando opportunamente il firmware.
È adatto alla risoluzione di reti combinatorie e rappresenta un ottimo supporto per il circuito **TinyPLC**, presentato nei numeri precedenti di Fare Elettronica.

Il software è Freeware ed è distribuibile su qualsiasi supporto o computer.
Può essere scaricato liberamente dal sito di Fare Elettronica.

GLI STABILIZZATORI DI TENSIONE

sesta parte

I REGOLATORI DI TIPO SWITCHING. I DC-DC CONVERTER

di Nico Grilloni
nicogrilloni@tin.it

Sono sempre più utilizzati prevalentemente per l'alto rendimento, notevolmente più elevato dei regolatori lineari fin qui analizzati e perché, se ben dimensionati, presentano un'eccezionale affidabilità.

Fino ad ora sono stati presi in considerazione gli stadi stabilizzatori di tipo lineare nei quali il componente regolatore in serie al carico, un BJT per esempio, lavora costantemente in zona attiva. Ciò comporta, come si è visto, una dissipazione di potenza non certo trascurabile per il BJT che, nella quasi totalità dei casi, dev'essere dotato di un robusto radiatore atto a dissipare all'esterno i relativi incrementi termici. Questo è essenzialmente il motivo per cui gli stadi stabilizzatori lineari non trovano applicazione per potenze maggiori di una cinquantina di watt presentando comunque, anche a potenze di qualche decina di watt, un rendimento mediamente non superiore al 40 o 50 %.

Per poter dissipare potenze più elevate è necessario ricorrere ai regolatori definiti di tipo switching nei quali il componente attivo, un BJT per esempio, non lavora più nella zona lineare della sua caratteristica, ma in commutazione, passando quindi alternativamente dalla zona di interdizione (stato OFF) alla zona di saturazione (stato ON), e viceversa, con una frequenza in genere compresa fra qualche decina e alcune centinaia di kHz. Ciò comporta un elevato rendimento del circuito (anche superiore al 90 %) dal momento che, come è noto, un BJT negli stati ON e OFF presenta una dissipazione minima. Nello stato di saturazione infatti, la corrente, comunque funzione del carico, è elevata ma è minima la caduta di tensione collettore-emettitore V_{CE} (dell'ordine di qualche decimo di volt). Nello stato di interdizione la V_{CE} approssima il valore della tensione di alimentazione ma la corrente I_c di collettore è prossima a zero. Pertanto, poiché è comunque valida la già nota espressione:

$$P_Q = (V_i - V_o)\, I_c = V_{CE}\, I_c$$

la potenza P_Q dissipata dal BJT sarà sempre modesta: nello stato ON di saturazione, infatti, è minima la V_{CE}, mentre nello stato di interdizione è minima (prossima a zero) la I_c di collettore. Il prodotto (V_{CE} x I_c) darà quindi sempre un risultato notevolmente modesto. Per effetto del basso valore dalla potenza dissipata dal BJT si può ritenere

che la potenza di ingresso venga quasi totalmente trasferita in uscita.
Un ulteriore pregio della tecnica switching risiede nella possibilità di realizzare i convertitori DC-DC che altro non sono che stabilizzatori in grado di realizzare circuiti sia di tipo step-down (o *buck*), ossia stadi in cui la V_o di uscita è inferiore alla V_i di ingresso, sia stadi stabilizzatori di tipo step-up (o *boost*), ossia stadi nei quali la V_o di uscita è maggiore della V_i di ingresso, sia stadi inverter (o *buck-boost*), ossia stadi nei quali la tensione V_o di uscita ha polarità opposta rispetto alla V_i applicata.
Unico neo dei regolatori switching risiede nell'elevato residuo in alternata che comunque può essere utilmente ridotto anche solo ricorrendo a più opportuni valori di alcuni componenti passivi.

SCHEMA DI PRINCIPIO

Lo schema di principio del regolatore switching è riportato nella figura 1 dove si suppone che il BJT passi dallo stato ON allo stato OFF e viceversa tramite un opportuno segnale V_P di pilotaggio inviato in base. In pratica il pilotaggio può eseguirsi tramite impulsi di frequenza fissa ricavati da un oscillatore o con impulsi di ampiezza costante e frequenza variabile o tramite un modulatore PWM (*Pulse Width Modulator*) che determina un opportuno segnale trigger.

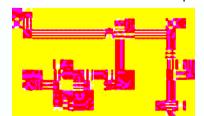



Figura 1: *Nei regolatori switching si ha un controllo dalla conduzione dell'elemento attivo (il BJT) in serie al carico tramite impulsi (Vp) generati da un apposito oscillatore*

Il controllo tramite PWM fa sì che nel caso in cui la tensione di uscita subisce, per esempio, un decremento ci sarà un'estensione dell'intervallo t_{ON} del segnale che comanda il passaggio in conduzione del componente attivo in serie al carico in modo che la minore caduta di tensione V_{CE} esegua la compensazione. Mentre, viceversa, nel caso in cui la tensione V_o subisce un incremento il medesimo intervallo t_{ON} del segnale di comando si contrarrà in modo che la conseguente maggiore caduta di tensione V_{CE} esegua l'opportuna compensazione.
Ciò implica che in ogni caso, al fine di mantenere stabile la tensione di uscita sia nei confronti delle sempre possibili fluttuazioni della tensione V_i di ingresso che nei confronti delle fluttuazioni del carico R_L, è comunque necessario inserire un circuito di reazione che, confrontando la tensione di uscita con una tensione di riferimento, determini l'opportuno segnale di pilotaggio della base del BJT in funzione delle variazioni della V_i e del carico R_L. La differenza fra i regolatori lineari e i regolatori switching è, a questo proposito, solo dovuta al fatto che nei primi si ha un controllo lineare del pilotaggio dell'elemento in serie al carico, mentre nei secondi il pilotaggio avviene tramite un oscillatore che produce un segnale variabile in ampiezza (ampiezza qui intesa come durata dell'impulso) o in frequenza
Ricordando che il *duty cycle* di un'onda quadra è dato dal rapporto:

$$\delta = t_{ON} / T$$

ossia dal rapporto fra l'intervallo di tempo in cui il segnale è a livello alto (t_{ON}) e il periodo T, si comprende come l'opportuno segnale di pilotaggio dovrà essere a *duty cycle* variabile. Dalla precedente espressione si vede che per ottenere la variabilità di δ, si può o mantenere costante l'intervallo t_{ON} variando la durata del periodo T o mantenere costante il periodo T variando la durata del t_{ON}.

IL CONVERTITORE DC-DC

La figura 2 riporta quindi il principio di funzionamento di uno stadio switching di tipo step-down. Qui la funzione di BJT è simulata dal deviatore S. Alla chiusura di questo la corrente, data la disposizione del diodo D, passa attraverso l'induttanza L e il carico R_C. L'induttanza si carica, quindi, tramite la resistenza R_C e la tensione ai suoi estremi è positiva al punto K e negativa all'estremo opposto (segni + e – disegnati all'esterno del circuito). Quando S cambia stato pas-

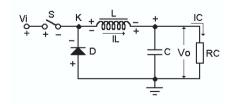



Figura 2: *Circuito a componenti passivi che simula un DC/DC converter di tipo step-down*

sando da chiuso ad aperto le polarità ai capi di *L* si invertono (segni – e + disegnati internamente all'area del circuito) e la stessa induttanza si comporta pertanto come un generatore che trasferisce l'energia immagazzinata al carico. La presenza del diodo, che adesso è polarizzato direttamente, consente alla corrente di fluire nel carico.
La frequenza *f* con cui il deviatore S si apre e si chiude, ovvero, con riferimento alla figura 3, la frequenza del segnale che pilota il BJT alternativamente in *ON* e in *OFF*, è in genere compresa, come si è detto, fra qualche decina e alcune centinaia di kHz. Si comprenderà come il ripple presente nella V_o di uscita sarà tanto minore quanto più elevata risulterà la frequenza *f*.
Per il convertitore di tipo step-down per la tensione V_o di uscita si ha l'espressione:

$$V_0 = V_i \times \frac{t_{ON}}{T}$$

Ma poiché t_{ON} / T, rapporto fra l'intervallo di tempo in cui il segnale di pilotaggio è a livello alto (t_{ON}) e il periodo $T = t_{ON} + t_{OFF}$ del medesimo segnale, è rappresentativo del duty cycle δ dello stesso segnale, l'espressione precedente può scriversi:

$$V_o = V_i \times \delta$$

Da questa si vede come la tensione di uscita dipenda dal duty cycle del segnale di comando (che è un numero minore di 1). Noto quindi δ si può immediatamente ricavare di quanto la tensione V_o di uscita è inferiore alla tensione V_i di ingresso.
La figura 3 riporta il medesimo circuito ma con il deviatore *S* sostituito da un BJT comandato in base dal segnale V_p.

Figura 3: *Il BJT Q o altro componete attivo (un Mosfet, per esempio) è il componente attivo che, in serie al carico, lavora in commutazione e che nella pratica sostituisce quindi il deviatore S di cui alla figura 2*

REGOLATORI SWITCHING INTEGRATI

Per realizzare stadi switching si ricorre ormai ai circuiti integrati per il cui funzionamento sono sufficienti pochi componenti esterni. La figura 4 espone, per esempio, uno stadio stabilizzatore di tipo switching step-down realizzato con l'IC LH1605 per il cui corretto funzionamento sono necessari soltanto una resistenza, un'induttanza e due condensatori. L'IC è posto in commercio in involucro TO-3 (lo stesso involucro, tanto per intenderci, del ben noto BJT di potenza 2N3055) nel cui interno vi è una sorgente per la tensione V_{ref} di riferimento compensata termicamente, un amplificatore di errore con relativo comparatore, un oscillatore a frequenza programmabile e un commutatore. Le caratteristiche dell'LH1605 sono:

- tensione di ingresso: (10 ÷ 35) V
- tensione di uscita: (3 ÷ 30) V
- corrente di carico: 5 A
- corrente di riposo: 20 mA (per I_L = 200 mA)
- dissipazione di potenza: 16 W (per V_o = 10 V e I_L = 5 A)
- regolazione di linea: 20 mV
- efficienza: 70 %
- resistenza termica R_{thj-c} = 5 °C / W

La figura 5 espone quindi il circuito analogico che semplifica il funzionamento dell'IC. Qui si vede un deviatore che, commutando alternativamente con frequenza *f* determinata dal condensatore C_T, dallo stato ON allo stato OFF e viceversa, connette e sconnette il carico R_L alla tensione di ingresso V_i. Ciò consente di modificare il ciclo di lavoro di un BJT in serie al carico (interno all'IC) in funzione della tensione presente al pin 3 (figura 4) che è connesso al positivo di uscita tramite la resistenza R_s. La tensione di errore, pari alla differenza fra la tensione presente al pin 3 e la tensione di riferimento presente al pin 2 viene comparata con un segnale a rampa prodotto da un oscillatore interno. Il risultato della compara-

Figura 4: *Stadio regolatore di tensione di tipo switching realizzato con l'IC LH1605*

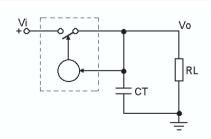



Figura 5: *Circuito a cui è riconducibile il funzionamento dell'IC LH1605 di cui alla figura 4*

zione determina un più o meno breve intervallo di permanenza del BJT in ON o in OFF. Il BJT pertanto, rimane in interdizione o in saturazione per il tempo necessario a compensare la variazione della tensione V_o di uscita.
Per il dimensionamento di uno stadio del tipo riportato nella figura 4 si inizia col determinare il valore da attribuire alla resistenza R_s per il quale si ricorrerà all'espressione:

$$R_s = \frac{2000 \times (V_0 - 2,5)}{2,5}$$

Poiché lo stadio di cui alla figura 4 è stato dimensionato per V_o = 5 V, dall'espressione precedente si ricava R_s = 2 kΩ.
Il valore da attribuire all'induttanza L si calcolerà con l'espressione:

$$L = \frac{(V_{imax} - V_0) \times V_0}{2 \times f \times I_{Lmin} \times V_{imax}}$$

Poiché lo stadio di cui alla figura 4 è stato dimensionato per V_{imax} = 18 V, V_o = 5 V, I_{Lmin} = 1 A e f = 27 kHz per l'induttanza L si ricava:

$$L = \frac{(18 - 5) \times 5}{2 \times 27000 \times 1 \times 18} = 66,87 \text{ mH}$$

Resta infine da calcolare il valore da attribuire al condensatore posto in uscita in parallelo al carico. Se si è in possesso di tutti i dati utili si applicherà l'espressione:

$$C > \frac{I_{Lmin}}{4 \times f} \times \frac{1}{V_r - (ESR \times I_{Lmin})}$$

dove V_r è la massima tensione di ripple voluta in uscita e ESR (*Effective Series Resistance*) è il termine che indica in sostanza le perdite nel condensatore che, come è noto, si rappresentano con una resistenza in serie al medesimo. In pratica, per la difficoltà di disporre di tutti i dati, si farà ricorso ad un condensatore elettrolitico al tantalio la cui capacità sarà di qualche migliaio di microfaraday avendo l'accortezza di disporvi in parallelo un condensatore ceramico da 0,1 µF al fine di ridurre i transitori nella commutazioni ON ÷ OFF e OFF ÷ ON.

CIRCUITO DI PROTEZIONE A LIMITAZIONE DI CORRENTE

La figura 6 riporta quindi la soluzione circuitale per realizzare un'efficace protezione dal sovraccarico per il convertitore di cui alla figura 4. Questa soluzione sfrutta la caratteristica dell'IC LH1605 per la quale si inibisce l'uscita una volta che sia portato a massa il pin 2. Nelle condizioni di normale funzionamento sono in interdizione entrambi i BJT Q_1 e Q_2. Se si verifica un incremento di corrente oltre il valore prestabilito, la caduta di tensione sulla resistenza R_s porta in conduzione il BJT Q_1 che, a sua volta, determina il passaggio in saturazione del BJT Q_2. Il collettore di quest'ultimo, e quindi il pin 2, si porta ad un potenziale assai prossimo al potenziale di massa e l'uscita dell'IC si porta in OFF. Indicando con I_{Lmax} il valore della corrente di carico al quale si vuole l'intervento della protezione, si calcolerà il valore da attribuire alla resistenza R_s con l'espressione:

Figura 6: *I BJT Q1 e Q2, unitamente alle due resistenze da 10 k e alla resistenza Rs, realizzano un'efficace protezione per lo stadio regolatore di cui alla figura 4*

$$R_s = 0,6 / I_{Lmax}$$

Un ulteriore e recente IC che consente di realizzare un ottimo regolatore step-down ad altissimo rendimento è l'LM2678 della National. Questo integrato è in grado di fornire una corrente di 5 A mantenendo un'ottima regolazione sia a fronte di variazioni della V_i che del carico R_L. Il componente è particolarmente versatile sia perché realizzato per tensioni fisse (3,3 V, 5 V e 12 V) e per tensioni variabili da 1,2 V a 37 V, sia perché utilizza induttanze di valore standardizzato e quindi di facile reperibilità. A tale proposito vale la pena ricordare come la pratica realizzazione dell'induttanza sia la nota dolente per lo sperimentatore che voglia costruire un buon alimentatore di tipo switching. La possibilità di utilizzare induttanze di valore standard non è quindi di secondaria importanza.
La serie LM2678 è dotata di shutdown termico e di un pin ON/OFF che consente la messa in standby del regolatore che, in questa condizione, assorbe una corrente (corrente di riposo)

non superiore a 50 μA. Il rendimento è maggiore del 90 % mentre la frequenza dell'oscillatore interno è pari a 260 kHz.

La figura 7 propone la versione a tensione V_o fissa e pari a 5 V per una corrente di carico di 5 A. Si noti come siano sufficienti solo pochi componenti esterni per la realizzazione di un affidabile *switcher*.
Si noti altresì la presenza del pin ON/OFF, pin 7, che, connesso a massa o comunque a un potenziale inferiore a 0,8 V, porta il regolatore in standby. La massima tensione da applicare al pin 7 per la condizione ON non deve eccedere i 6 V. Nel caso in cui il comando ON/OFF non si utilizzi, si lascerà sconnesso il pin relativo.
La capacità complessiva posta in ingresso (3 x C_i) è di 45 μF mentre la capacità complessiva dei condensatori C_o posti in uscita è pari a 360 μF. Il ricorso a più condensatori in parallelo piuttosto che ad un singolo condensatore di eguale capacità consente di pervenire ad una minore ESR.
La presenza del diodo Schottky, diodo veloce a bassa caduta di tensione anodo-catodo, è necessaria per fornire una via di estinzione alla corrente che continua a fluire nell'induttanza quando il regolatore viene posto in OFF (intendendo per stato OFF, in questo caso, la disinserzione dell'alimentazione V_i).

La figura 8, infine, mostra la configurazione a cui fare riferimento per ottenere in uscita una qualsiasi tensione V_o compresa fra 1,2 V e 37 V. Si può constatare come questo regolatore si differenzi dal precedente a tensione fissa solo per la presenza delle due resistenze R_1 e R_2 il cui valore, unitamente al valore della tensione di reazione V_{FB} (per il quale la Casa impone il valore di 1,21 V), determina appunto la V_o desiderata. Per quest'ultima si ha infatti:

$$V_0 = V_{FB} \times \left(1 + \frac{R_2}{R_1}\right)$$

Fissato il valore della resistenza R_1 (la Casa suggerisce R_1 = 1 kΩ) si calcolerà il valore della R_2 con l'espressione:

$$R_2 = R_1 \times \left(\frac{V_0}{V_{FB}} - 1\right)$$

CONSIDERAZIONE CONCLUSIVA

Si sono già poste in evidenza le differenze più significative fra i regolatori lineari e i regolatori switching. In conclusione si vuole soltanto evidenziare come i circuiti integrati atti alla realizzazione dei secondi presenti oggi sul mercato della componentistica attiva, abbiano notevolmente semplificato i relativi criteri di dimensionamento. Le Case costruttrici, infatti, oltre ad aver realizzato IC a configurazione circuitale anche notevolmente sofisticata ma di facile applicazione (si è già fatto notare come siano sufficienti pochi componenti passivi all'esterno dell'IC per un ottimale funzionamento del medesimo), sono solite fornire dati, diagrammi, formule e utili suggerimenti (anche per il cablaggio) che conducono rapidamente alla realizzazione di un regolatore switching. Ben diverso è il discorso nel caso si voglia eseguire il dimensionamento di questi circuiti con componenti discreti ai quali restiamo affezionati ma che, nella fattispecie, sono ormai da non tenersi in alcuna considerazione.
Si fa infine presente che qui, per ovvi motivi, si è fatto riferimento, nel trattare i convertitori DC-DC, solo al tipo step-down. Si rimane comunque a disposizione del Lettore per qualunque quesito inerente i regolatori step-up e buck-boost e per qualsiasi ulteriore approfondimento.

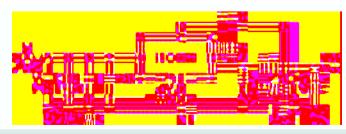



Figura 7: *Stadio regolatore switching realizzato col versatile IC LM2678 nella versione per tensioni fisse*

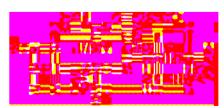



Figura 8: *Stadio regolatore switching realizzato con l'IC LM2678 nella versione per tensioni variabili*

CONVERTITORE PER LAMPADE FLUORESCENTI

di Marco Lento
sglent@tin.it

Pochi componenti di facile reperibilità per costruire un semplice convertitore utile ad accendere con 12 Volt i normali tubi fluorescenti da 36 W; ideale per il campeggio o in casa come complemento di sistemi anti black-out.

In più di una situazione può risultare utile accendere una lampada al neon da 36 W disponendo di una tensione di soli 12 Volt; si pensi ad esempio alle esigenze del campeggiatore oppure all'allestimento di una luce di emergenza per la casa.

Malgrado negli ultimi anni, grazie alla tecnologia PWM, il rendimento degli inverter a 50 Hz sia notevolmente aumentato (anche il 90 % della potenza assorbita viene reso in uscita), servirsi di simili apparati solo per accendere una lampada fluorescente appare superfluo; meglio realizzare un semplice dispositivo come quello che passo a descrivere.

LO SCHEMA

Il circuito sfrutta la capacità dei tubi fluorescenti di accendersi istantaneamente, quando si applica sui catodi dell'alta tensione ad una frequenza più elevata dei 50 Hz di rete. Questo principio, largamente utilizzato nelle lampade portatili, consente l'accensione dei tubi senza fare uso di reattore e starter, indispensabili se li si alimenta a 220V-50Hz.

I tre transistori visibili nello schema formano quindi un semplice generatore di alta tensione. Il multivibratore astabile TR1-TR2 fornisce un treno di onde quadre alla frequenza di circa 600 Hz; questo segnale, applicato sulla base del Darlington TR3, può pilotare il primario di T1, un

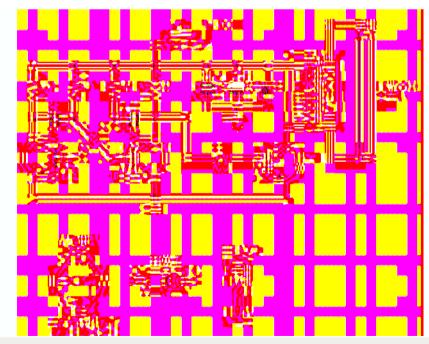



Figura 1: *Schema elettrico del convertitore*

normale trasformatore 9V-2A utilizzato alla rovescia. In virtù dell'elevata frequenza di funzionamento, la tensione prelevabile sarà dell'ordine degli 800 V a vuoto, che scenderanno a circa 300 V subito dopo l'innesco della lampada.
Il circuito è completato da D1 che protegge l'intero modulo da accidentali inversioni di polarità, interrompendo il fusibile posto a monte del cordone di alimentazione; l'assorbimento del circuito è dell'ordine dei 2 A e può essere variato modificando il valore di R5 alla ricerca del miglior compromesso tra consumo, facilità di innesco e luminosità della lampada.

COSTRUZIONE ED UTILIZZO

Realizzato il circuito stampato, il montaggio dei pochi componenti è dei più semplici. Una nota merita il finale TR3 che deve essere adeguatamente raffreddato, applicandolo su un dissipatore di generose dimensioni.
Riguardo al trasformatore, qualsiasi modello con nucleo lamellare o toroidale si presta allo scopo; per alimentare i catodi della lampada conviene servirsi di due fili dalla sezione anche modesta ma non appaiati; è inoltre possibile l'accensione di più tubi di potenza minore, collegandoli tra di loro in serie.
Approntato il cavo di alimentazione, dotato a monte di fusibile da 3A, si può procedere al collaudo del modulo; in assenza di punti di taratura, il funzionamento, accompagnato da un acuto sibilo proveniente dal trasformatore, deve essere immediato. L'alta tensione presente in uscita, suggerisce di alloggiare il tutto in una plafoniera di materiale plastico; ricordo inoltre di non collegare al convertitore apparati funzionanti a 220 V, pena la loro distruzione.
Attenzione alle tensioni in gioco che possono essere pericolose, quindi utilizzare un contenitore plastico e ponete molta attenzione nel maneggiare il circuito.

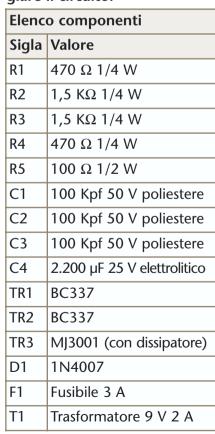
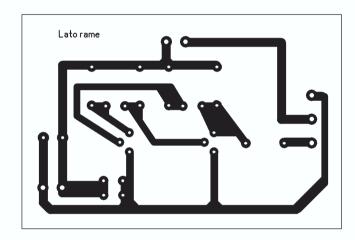



Figura 2: *Circuito stampato scala 1:1 (lato rame)*

Figura 3: *Posizionamento componenti*

Elenco componenti	
Sigla	**Valore**
R1	470 Ω 1/4 W
R2	1,5 KΩ 1/4 W
R3	1,5 KΩ 1/4 W
R4	470 Ω 1/4 W
R5	100 Ω 1/2 W
C1	100 Kpf 50 V poliestere
C2	100 Kpf 50 V poliestere
C3	100 Kpf 50 V poliestere
C4	2.200 µF 25 V elettrolitico
TR1	BC337
TR2	BC337
TR3	MJ3001 (con dissipatore)
D1	1N4007
F1	Fusibile 3 A
T1	Trasformatore 9 V 2 A

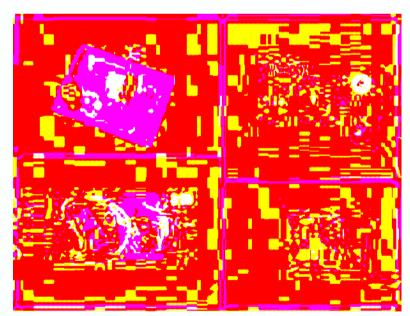
CARICABATTERIE NI-CD/MH A 3 PORTATE CON LM 317

di Marco Lento
sglent@tin.it

Con un integrato LM317 e pochi altri componenti costruiamo un caricabatterie a corrente costante per elementi nickel-cadmio e nickel-metal hydride.

L'utilizzo di batterie ricaricabili negli apparati portatili è oramai di uso comune. Agli indubbi vantaggi di tipo economico, si sommano i benefici ecologici e quelli legati alle migliori prestazioni elettriche ottenibili; se gli elementi NI-CD erano in grado erogare picchi di corrente impensabili per le normali pile zinco-carbone, tali caratteristiche sono ulteriormente migliorate con la tecnologia NI-MH (Metal-Hydride) con la quale, oltre ad evitare il fastidioso effetto memoria tipico delle NI-CD, si ottengono capacità più elevate che possono raggiungere già per il formato "stilo" AA i 2300 mA. In ogni caso, per procedere alla ricarica degli elementi NI-CD e NI-MH, occorre disporre di un caricatore a corrente costante, come quello che passo a descrivere.

LO SCHEMA

Improntato alla massima semplicità, il circuito proposto si basa su un integrato LM317. Il componente, più noto come stabilizzatore di tensione, si trasforma in generatore di corrente costante collegando tra i terminali R (regolazione) ed U (uscita) un resistore di adeguato valore, consentendoci di prelevare dal terminale R la corrente richiesta.

La formula per calcolare il valore de questa resistenza è la seguente:

R (in Ω) = 1250 : mA uscita

I valori così ottenuti potranno essere arrotondati senza problemi al più vicino valore standard; ricordo di non oltrepassare gli 1,5 Ampere, massima corrente erogabile da un LM 317 correttamente raffreddato. Il commutatore S2 consente la selezione di 3 portate di corrente (terminali C, 1, 2 e 3) mentre il circuito è completato dall'indicatore di ricarica in corso (DL 2); questo LED, alimentato dalla differenza di potenziale introdotta da D1, D2, e D3, si accenderà soltanto ad elementi correttamente collegati.

Il trasformatore di alimentazione consigliato (24 V 0,5 A) consente di collegare in uscita fino a 10 elementi da porre in serie.

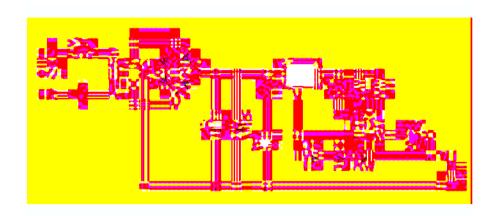



Figura 1: Schema elettrico del caricabatterie

Elenco componenti	
Sigla	**Valore**
R1	2,7 KΩ 1/2 W
R2	82 Ω 1/2 W
R3	4,7 Ω 2 W
R4	2,7 Ω 5 W (a filo)
R5	47 Ω 1/4 W
C1	1000 μF 50 V elettrolitico
C2	100 Kpf 50 V poliestere
IC1	LM317T
D1	1N4007
D2	1N4007
D3	1N4007
DL1	LED 3 mm
DL2	LED 3 mm
S1	Interruttore
S2	Commutatore 1via 3 posizioni
T1	Trasformatore 24 V 0,5 A
P1	Ponte 4 Ampere
F1	Fusibile 300 mA

COSTRUZIONE ED UTILIZZO

Realizzato il circuito stampato qui proposto, il montaggio dei pochi componenti tutti di facile reperibilità è dei più semplici. Ricordo di raffreddare adeguatamente l'integrato LM317 che, qualora si colleghino in uscita soli 2-4 elementi, sarà chiamato a dissipare, per caduta di tensione, una discreta quantità di calore.
Prima di procedere alla ricarica di elementi NI-CD è necessario provvedere alla loro scarica completa per annullare l'effetto memoria che ne ridurrebbe notevolmente l'autonomia; questa operazione, che può essere effettuata tramite una semplice lampadina, non è invece richiesta per le più recenti NI-MH.
Le portate 450 mA (terminali C e 1) e 260 mA (terminali C e 2) sono utilizzabili per la ricarica in 5/8 ore di pile formato AA da 2100 mA, mentre la portata 15 mA (terminali C e 3) risulta utile per elementi di bassa capacità che accettano solo la modalità di ricarica lenta, come quelli impiegati nei telefoni cordless.
Per un migliore trattamento delle pile ricordo di azionare il commutatore S2 solamente a caricatore spento.

Buona ricarica!

Figura 2: Circuito stampato scala 1:1 (lato rame)

Figura 3: Piano di montaggio

FEBOT terza parte

IL FIRMWARE DELLA SCHEDA MADRE

di ***Maurizio Del Corso*** *(m.delcorso@farelettronica.com)*
e ***Tiziano Galizia*** *(t.galizia@farelettronica.com)*

Il mese scorso abbiamo sviscerato il progetto hardware della scheda madre e questo mese analizzeremo il firmware nel dettaglio in modo che possiate adattarlo ai vostri progetti nella maniera più semplice possibile. Verranno descritte le routines facendo largo uso dei diagrammi di flusso in modo che sia chiara la loro funzione e possano essere "tradotte" facilmente anche in altri linguaggi di programmazione. Visto il livello di approfondimento, l'analisi del firmware verrà completata con il prossimo numero anche se fin da adesso potrete scaricare il codice completo sul nostro sito Internet.

UN FIRMWARE IN CONTINUA EVOLUZIONE

Come in tutti i dispositivi elettronici intelligenti, il firmware è una componente in continua evoluzione ed anche per la scheda madre del FEbot non ci sono eccezioni. Benché il firmware che ci accingiamo a descrivere sia testato e funzionante, solo utilizzando la scheda madre con altre schede nelle diverse configurazioni potranno emergere eventuali miglioramenti e revisioni, che vi invitiamo comunque a proporre, effettuare e segnalarci in modo che il sistema sia sempre operativo nel modo più ottimale.

Il firmware è stato scritto in Assembler e questo anche se complica un po' la comprensione, consente la modifica e la ricompilazione senza disporre di particolari strumenti software. Per la compilazione è sufficiente disporre del compilatore MPASM che è parte del pacchetto software gratuito MPLAB di Microchip che potete scaricare gratuitamente dal sito Microchip, ma che trovate anche all'interno del CDROM allegato a Fare Elettronica n.231 (Settembre 2004) o anche nel CDROM del corso "*PICmicrocontroller® by example*".

Una comprensione del firmware nei minimi dettagli, vi darà la possibilità di utilizzare il codice adattandolo al vostro micro anche se diverso dal PIC16F876 utilizzato sulla scheda madre. La versione del firmware a cui faremo riferimento in questo articolo è la prima assoluta: *V.1.0 – ottobre 2004*.

Inutile dire che contiamo sulla vostra collaborazione per il rilascio delle versioni successive!

SEMPLIFICHIAMO LE COSE

Affinché il firmware sia comprensibile, è necessario che sia strutturato in maniera ben defi-

nita in modo da creare una sorta di standard per la stesura del codice. In questo modo, anche se cambiano i contenuti, la struttura rimane sempre la stessa, facilitando la comprensione del codice anche se si analizzano programmi scritti da altri utenti. Per fare questo utilizzeremo la direttiva *INCLUDE nomefile* la quale fa sì che in fase di compilazione venga inserito il file *nomefile* nella posizione occupata dalla direttiva INCLUDE.

Chi progetta un modulo del FEbot deve creare due file: *main.asm* e *var.inc.* Il file *main.asm* contiene il programma principale e la sua struttura è riportata in figura 1. Si noti che in *main.asm* vengono inclusi i file *variabili.inc* e *library.inc* che vengono forniti da Fare Elettronica e contengono le routines di gestione del bus secondo le specifiche e le relative dichiarazioni delle variabili. *variabili.inc* a sua volta include il file *var.inc* che è il file, prodotto dagli utenti, contenente la

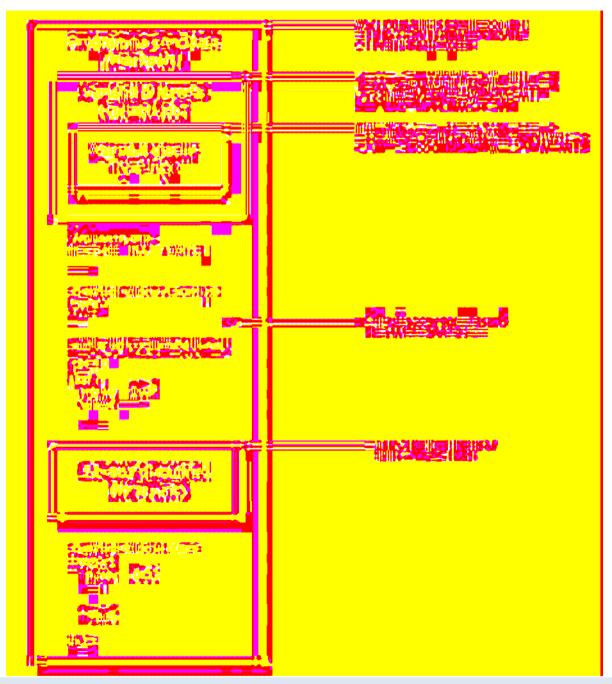



Figura 1: *: Struttura del programma assembler*

dichiarazione delle variabili necessarie alle routines create dai lettori. Sul sito di Fare Elettronica sono disponibili i modelli dei vari file e, ovviamente, *library.inc* e *variabili.inc.*
Dunque, come tutti gli altri moduli, anche per la scheda madre si deve scrivere il file *main.asm* e *var.inc.* Sul file *var.inc* non ci soffermeremo a lungo in quanto contiene la dichiarazione delle variabili ed alcune definizioni. Le definizioni risultano particolarmente utili in quanto permettono di associare dei nomi logici a costanti o ai pin fisici delle porte del PIC: in questo modo possiamo utilizzare tali nomi nel programma principale ed eventuali modifiche possono essere effettuate cambiando semplicemente le definizioni e lasciando inalterato il codice del programma.
Consigliamo sempre di inserire un gran numero di commenti nel codice (ricordiamo che i commenti sono preceduti da un punto e virgola) al fine di rendere il programma leggibile e comprensibile anche da chi non ha partecipato alla stesura.
Nell'analisi del codice che inizieremo nel prossimo paragrafo, faremo largo uso dei diagrammi di flusso in modo da capire quali sono le operazioni principali eseguite dalle varie routines. Faremo riferimento alle linee di codice assembler solo quando strettamente necessario, al fine di non appesantire troppo la trattazione.

IL FILE 'VAR.INC'

Come già accennato, il file var.inc contiene le dichiarazioni delle variabili definite dall'utente e alcune definizioni utili per semplificare la comprensione del codice. Nel riquadro che segue sono riportate le definizioni contenute nel file *var.inc*:

La prima parte delle definizioni riguarda i segnali elettrici, quindi ad esempio l'istruzione *bsf ERR* per accendere il led ERROR, sarà equivalente a *bsf PORTC,5.*
La seconda parte è relativa ai dati tipici della scheda madre, come l'indirizzo, il codice di identificazione, i comandi riconosciuti, ecc. Le definizioni sono particolarmente utili: se ad esempio volessimo cambiare l'indirizzo ADDR da 01H a 02H, basterebbe associare 02H alla costante ADDR con la riga *#DEFINE ADDR 02H* e lasciare invariato tutto il resto del programma.

IL PROGRAMMA PRINCIPALE

Un ciclo infinito…

Il programma principale inizia all'etichetta *start* e, come prima cosa, configura i pin di I/O del PIC come ingressi o come uscite a seconda della funzione assegnata. In particolare vengono impostati per la PORTA RA1 come uscita e gli altri come ingressi, la PORTB come ingressi, mentre per la PORTC viene impostato RC7 ed RC1 come ingresso e gli altri come uscite. Vengono inoltre abilitate le interruzioni, delle quali parleremo più avanti. In realtà, come mostra la figura 2, il programma principale è un ciclo continuo (l'inizio del ciclo è contrassegna-

VAR.INC

```
;definizioni dei segnali della scheda madre
#DEFINE LOWBAT   PORTA,1    ;LED segnalazione batt. scarica
#DEFINE ERR      PORTC,5    ;LED segnalazione errori di comunicazione
#DEFINE RX       PORTC,7    ;linea ricezione RS232
#DEFINE TX       PORTC,6    ;linea trasmissione RS232
#DEFINE VBAT     PORTA,0    ;linea di monitoraggio della tensione di alimentazione

;indirizzo della scheda madre e comandi riconosciuti
#DEFINE ADDR     0x01       ;l'indirizzo della scheda madre è 00000001
#DEFINE MM       0x0A       ;mese di convalida (fornito da FE)
#DEFINE AA       0x04       ;anno di convalida (fornito da FE)
#DEFINE POWERON  0XFF       ;Comando di accensione (standard)
#DEFINE STANDBY  0X00       ;Comando di standby (standard)
#DEFINE RESET    0X01       ;Comando di reset (standard)
#DEFINE ID       0XFF       ;Comando di identificazione (standard)
#DEFINE L_BAT    0XFF       ;Comando di batteria scarica (standard)
#DEFINE PC_CONN  0XFF       ;Comando di notifica connessione PC (standard)
```

to dall'etichetta *ciclo1*) che compie le seguenti azioni:

- Analisi della linea BUSY_S. Prosegue solo se BUSY_S è a livello alto.
- Ricezione dei dati (mediante la routine *Ricevi_CMD*).
- Controllo della correttezza dei dati ricevuti: in caso negativo si torna all'analisi della linea BUSY_S.
- Analisi dell'indirizzo del pacchetto dati ricevuto: se è il proprio (per la scheda madre è 01H) o è broadcast (FFH) si analizza il comando altrimenti si torna all'analisi della line BUSY_S.
- Analisi ed esecuzione del comando.
- Torna all'inizio del ciclo.

Vediamo in dettaglio in che modo avviene il controllo dell'indirizzo di destinazione del pacchetto ricevuto. Nell'ipotesi che il pacchetto sia stato ricevuto correttamente, nella variabile *indir* si troverà l'indirizzo che dobbiamo analizzare. La figura 3 riporta il frammento di codice che esegue questo controllo. Per prima cosa viene controllato se l'indirizzo è broadcast (FFH) quindi un pacchetto destinato a tutti i moduli. Per eseguire questo controllo viene incrementato di uno il valore di *indir* mediante l'istruzione *incfsz indir*,0. Questa istruzione incrementa il contenuto di *indir* memorizzando il risultato in *w* e l'istruzione successiva viene eseguita solo se il risultato è diverso da zero. Se *indir* contiene il valore FFH, l'incremento azzererà *w* e verrà eseguita l'istruzione *goto si_addr* che provoca un salto alla sezione di codice dedicata all'analisi del comando. Se indir non contiene FFH, l'incremento darà un risultato non nullo per cui verrà eseguita l'istruzione *goto no_broad* che salta alla sezione di codice dedicata all'analisi dell'indirizzo. L'analisi dell'indirizzo avviene confrontando il valore di *indir* con la costante *ADDR* alla quale, nella sezione di definizione, è stato associato l'indirizzo della scheda madre (01H). Il confronto viene fatto memorizzando *indir* in *w* e sottraendoci il valore *ADDR*: se il risultato è zero il pacchetto era destinato proprio alla scheda madre, altrimenti si torna in ascolto della BUSY_S. Ricordiamo che quando una istruzione dà zero come risultato, viene portato ad 1 il bit Z del registro STATUS (noto appunto come *Zero Flag*).
La stessa tecnica viene usata per l'analisi del comando, confrontando la variabile *cmd* con i vari

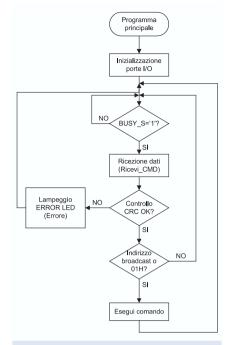



Figura 2: *Il diagramma di flusso del programma rincipale*

```
   Incfsz indir,0;     controllo se indir=ff (broadcast)
   goto   no_broad
   goto   si_addr      ;ricevuto un pacchetto broadcast
no_broad
   movf   indir,w      ;metto in w il campo indirizzo del pacchetto ricevuto
   sublw  ADDR         ;non era un pacchetto broadcast, allora
   btfsc  STATUS, Z    ;controllo se l'ind ricevuto è quello della scheda madre
(01H)
   goto   si_addr
no_addr
   goto   ciclo1       ;il pacchetto non era destinato alla scheda madre
                       ;- torna all'inizio del ciclo
si_addr
   ;sezione di analisi del comando
   ...
```

Figura 3: *Struttura del programma assembler*

comandi supportati (si veda il diagramma di flusso di figura 4). I comandi supportati sono memorizzati nelle relative costanti e, per la scheda madre, sono: RESET e ID. Il primo provoca il reset della scheda riportando il programma al punto di partenza, mentre il secondo provoca l'identificazione della scheda. Per la scheda madre i parametri di identificazione (6 byte) sono memorizzati nelle costanti ADDR (=01H), MM (=0AH), AA (=04H) e gli altri 3 byte sono zero.

Le interruzioni

La scheda madre ha il compito di monitorare il livello della tensione di alimentazione e notificare eventualmente lo stato di batteria scarica. Per fare questo, viene acquisito il valore della tensione di alimentazione ogni minuto e se questo è inferiore a circa 4,7V viene inviato un pacchetto broadcast di notifica,

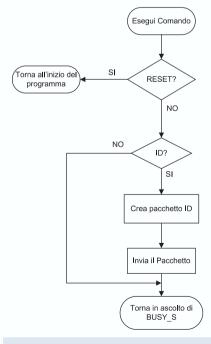



Figura 4: *Analisi ed esecuzione del comando*

contenente tale valore. Il primo problema è dunque capire quando è trascorso un minuto.
Per questo si utilizza la periferica Timer0. Con il quarzo da 4MHz e tenendo conto che l'esecuzione di una istruzione occupa 4 cicli di clock, Timer0 effettuerà 1000000 di conteggi al secondo. Ogni volta che Timer0 raggiunge il suo valore massimo (256 conteggi) genera una interruzione, per cui avremo 1000000/256=3906,25 interruzioni prima che sia trascorso un secondo.
Le variabili TimerH:TimerL vengono incrementate ad ogni interruzione per cui quando il loro valore raggiunge 3906,25 siamo sicuri che è trascorso un secondo. Poiché le variabili TimerH:TimerL non possono contenere valori decimali, possiamo assumere, con buona approssimazione, che sia trascorso un secondo quando il conteggio raggiunge il valore 3907 (=0F43H). Una volta raggiunto tale valore, viene incrementata la variabile *secondi* e solo quando *secondi* raggiunge il valore 60, viene letta la tensione di alimentazione ed azzerato il conteggio dei secondi. Il diagramma di flusso della routine di interruzione relativa al Timer0 è riportato nella figura 5.

Vediamo ora in che modo avviene la lettura della tensione di alimentazione. Il conduttore di alimentazione è stato collegato alla porta RA0 del PIC che viene configurato come ingresso analogico. Per gestire la conversione AD dobbiamo fare uso dei registri ADCON1 e ADCON0 del PIC. Il risultato della conversione verrà memorizzato nei registri ADRESH:ADRESL. In figura 6 è riportato il codice della routine *Leggi_V* che opera appunto la lettura della tensione.

Come si può vedere, per prima cosa vengono azzerati ADRESH:ADRESL che potrebbero contenere eventuali risultati di precedenti conversioni, quindi viene caricato il valore 10001110 nel registro ADCON1. Il bit più significativo (l'1 più a sinistra) è il bit ADFM ed indica che il risultato della conversione sarà allineato a destra. Questo perché il convertitore AD del PIC è a 10 bit, mentre il risultato viene memorizzato in ADRESH:ADRESL che sono due registri ad 8 bit. Allineando il risultato a destra, i 6 bit più significativi di ADRESH saranno posti tutti a 0 (figura 7).

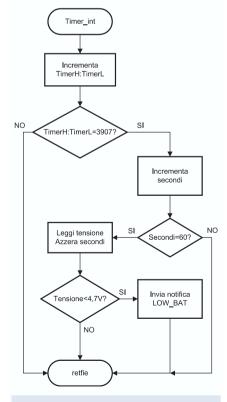



Figura 5: *Interruzione del Timer0*

I successivi tre bit a 0 di ADCON1 non sono significativi, mentre gli ultimi quattro bit di ADCON1 selezionano il canale analogico RA0 con tensioni di riferimento +Vcc e GND.

Successivamente viene inizializzato il valore di ADCON0 mettendo a zero i bit 3, 4 e 5 (selezione del canale RA0), mettendo i valori 1 e 0 rispettivamente nei bit 6 e 7 (imposta la frequenza di campionamento a Fosc/8 quindi 500KHz) quindi viene attivato il convertitore ponendo ad 1 il bit 0.
La routine *delay160* provoca un ritardo di 160 microsecondi per consentire il completamento dell'acquisizione del valore da convertire quindi ponendo ad 1 il bit 2 di ADCON0 si da inizio all'operazione di conversione. Quando la conversione è terminata il bit 2 di ADCON0 viene riportato automaticamente a zero per cui la routine entra in un ciclo in cui esamina il bit 2 di ADCON0 ed esce solo quando tale bit vale 0.
Terminata la conversione viene disattivato il convertitore AD riportando a zero il bit 0 di ADCON0.

Come avviene i controllo del valore della tensione di alimentazione? Come già detto il convertitore AD del PIC16F876 è a 10 bit e, poiché sono state scelte come tensioni di riferimento Vcc e GND, significa che il range di 5V consentito per la conversione, viene suddiviso in 2^{10}=1024 parti. La minima risoluzione è dunque pari a 5/1024 ovvero circa 4,88mV. Se la tensione di alimentazione vale 4,7V il risultato della

```
Leggi_V
    Clrf    ADRESH
    bsf     STATUS,RP0
    clrf    ADRESL
    bcf     STATUS,RP0
    bsf     STATUS,RP0
    movlw   B'10001110'
    movwf   ADCON1
    bcf     STATUS,RP0
    bcf     ADCON0,3
    bcf     ADCON0,4
    bcf     ADCON0,5
    bsf     ADCON0,6
    bcf     ADCON0,7
    bsf     ADCON0,0
    call    delay160
    bsf     ADCON0,2
    nop
    btfsc   ADCON0,2
    goto    $-2
    call    delay160
    call    delay160
    bcf     ADCON0,0
    return
```

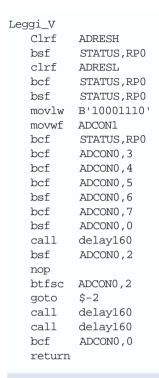
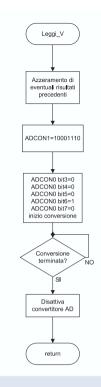



Figura 6: *Lettura della tensione di alimentazione*

conversione sarà dato da :

4,7 * 1024/5=962,56

Possiamo assumere dunque che se ADRESH:ADRESL contiene 962 (in binario 00000011:11000010) la tensione vale 4,7V. La routine *Timer_int* provvede a fare questa verifica e se il valore in ADRESH:ADRESL è inferiore a 962, invia un messaggio broadcast di batteria scarica contenente il valore della tensione di alimentazione. Questo messaggio potrebbe essere interpretato dagli altri moduli eventualmente presenti, provocandone lo spegnimento oppure da un modulo di visualizzazione che potrebbe visualizzare un messaggio particolare su un display LCD o su un display led a 7 segmenti.
Non abbiamo ancora esaminato in che modo si risale all'evento che ha scatenato l'interruzione. Ricordiamo che le interruzioni possono essere provocate da

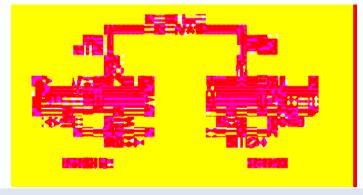



Figura 7: *Allineamento del risultato nei registri ADRESH:ADRESL*

diversi eventi che si verificano internamente o esternamente al microcontrollore. Per prima cosa è dunque opportuno abilitare solo le interruzioni che ci servono. Esiste un registro per la gestione delle interruzioni, mediante il quale è possibile sia abilitare o meno determinate periferiche alla generazione di interrupt, sia verificare quale periferica ha scatenato l'interruzione. Il registro è INTCON e la sua struttura è quella di figura 8.

GIE consente di abilitare (se ad 1) o meno le interruzioni in senso globale. PEIE abilita o meno le interruzioni per le periferiche, TMR0IE abilita timer0 ad richiedere interruzioni, mentre INTE e RBIE abilitano le interruzioni esterne ed in particolare INTE quelle su RB0/INT mentre RBIE quelle sul cambiamento di stato di uno dei bit sulla PORTB. I tre bit meno significativi sono i flag di interruzione che permettono di risalire alla periferica che ha generato l'interruzione. Questi falgs vengono messi ad 1 al momento dell'interruzione e devono essere azzerati via software al momento in cui viene servita l'interruzione.
Quando si scatena una interruzione, il programma salta automaticamente all'indirizzo 04H al quale deve trovarsi il codice di gestione delle interruzioni. È fondamentale salvare lo stato dei registri prima di servire l'interruzione, in modo da ripristinare il tutto una volta terminata la routine di servizio. Per questo motivo all'indirizzo 04H (individuabile dalla direttiva ORG 04) viene, come prima cosa, salvato il valore di *w*, del registro STATUS e del registro PCLATH rispettivamente nelle variabili *w_safe*, *status_safe* e *pclath_safe*. Questi valori verranno successivamente ripristinati dalla routine di interruzione. Successivamente vengono analizzati i flags di interruzione per capire chi ha scatenato l'evento, quindi saltare alla routine di servizio opportuna. In figura 9 è rappresentato un diagramma di flusso per l'analisi delle interruzioni. A titolo di esempio sono riportate diverse interruzioni che non necessariamente saranno usate nel nostro codice.

Lasciamo alla prossima puntata l'analisi delle routine di lettura e scrittura di un pacchetto sul bus e quelle di gestione della comunicazione con un PC mediante la porta seriale. Ci occuperemo anche di come interpretare i dati inviati al PC e come interrogare la scheda madre sfruttando al porta seriale RS232.

Figura 8: *Struttura del registro INTCON*

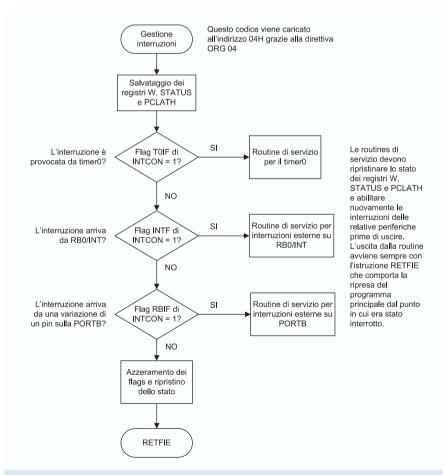



Figura 9: *Gestione delle interruzioni: un esempio con interruzioni di timer0, esterne su RB0/INT e su cambiamento di stato di un pin della PORTB.*

NUOVO CORSO

GLI ALIMENTATORI SWITCHING

Grazie a particolari tecniche, gli alimentatori switching possono raggiungere rendimenti altissimi se paragonati agli alimentatori lineari.

Per questo motivo gli switching trovano largo impiego nell'elettronica di tutti i giorni: l'alimentatore del PC, il caricabatteria del cellulare, del palmare, del notebook...

Ma quali sono le tecniche utilizzate negli switching?
Quali configurazioni esistono e quali caratteristiche presentano?

A queste ed altre domande potrete trovare risposta seguendo il nostro corso dedicato agli alimentatori switching che, oltre alla parte teorica, presenterà una serie di progetti pratici per concretizzare ciò che avrete appreso, ed alla fine...

gli alimentatori switching per voi non avranno più segreti!

A partire da DICEMBRE 2004

INDICE DELLE USCITE

PRIMA PARTE (FE-234)
- Introduzione;
- Alimentatori lineari: cenni
- Alimentatori switching, charge-pump e magnetici;
- Tipologie di switching: la tipologia boost;
- Progetto n.1: switching di tipo boost senza regolazione.

SECONDA PARTE (FE-235)
- Tipologie di sistemi di controllo: la tipologia PFM;
- Perdite negli switching: generalità;
- Perdite nei condensatori;
- Progetto n.2: sistema di controllo PFM per switching boost.

TERZA PARTE (FE-236)
- Perdite nei MOSFET;
- Tipologie di switching: la tipologia buck;
- Un circuito integrato PFM commerciale: l'MC34063;
- Progetto n.3: switching di tipo buck ad alta corrente con controllo PFM.

QUARTA PARTE (FE-237)
- Perdite nei diodi;
- MC34063: analisi dettagliata;
- Tipologie di switching: la tipologia inverting;
- Progetto n.4: switching di tipo inverting con controllo PFM.

QUINTA PARTE (FE-238)
- Perdite negli induttori: prima parte;
- Tipologie di sistemi di controllo: la tipologia PWM Voltage-Mode;
- Tipologie di switching: la tipologia buck-boost;
- Progetto n.5: switching di tipo buck-boost con controllo PWM V-Mode.

SESTA PARTE (FE-239)
- Circuiti integrati PWM commerciali: SG3524/5;
- Isolamento ingresso-uscita negli switching: trasformatori magnetici;
- Materiali ferromagnetici;
- Perdite magnetiche negli induttori e nei trasformatori.

SETTIMA PARTE (FE-240)
- Tipologie di switching a trasformatore: la tipologia forward single-ended;
- Tipologie di switching a trasformatore: la tipologia forward push-pull;
- Progetto n.6: switching di tipo forward push-pull con controllo PWM V-Mode.

OTTAVA PARTE (FE-241/242)
- Tipologie di sistemi di controllo: la tipologia PWM Current-Mode;
- Circuiti integrati PWM commerciali: famiglie UC380x e UCC380x.

NONA PARTE (FE-243)
- Tipologie di switching a trasformatore: la tipologia flyback;
- Progetto n.7: switching di tipo flyback ad uscita singola con controllo PWM C-Mode;
- Il problema dell'isolamento ingresso-uscita;
- Isolamento tramite trasformatori;
- Isolamento tramite optoisolatori.

DECIMA PARTE (FE-244)
- Switching multiuscita: generalità;
- Switching multiuscita: considerazioni avanzate;
- Progetto n.8: switching di tipo flyback ad uscita multipla con controllo PWM C-Mode.

UNDICESIMA PARTE (FE-245)
- Alimentatori switching da rete: generalità;
- Alimentatori switching da rete: sicurezza elettrica;
- Alimentatori switching da rete: stadi di ingresso.

DODICESIMA PARTE (FE-246)
- Alimentatori switching da rete: stadi di potenza;
- Alimentatori switching da rete: stadi di uscita;
- Cenni su Power Factor Correction.

ASPIRONE: terza parte

NON PERDE LA BUSSOLA

di Marco Fabbri
marnic@roboitalia.com

Dotiamo il nostro robot aspirapolvere di una bussola elettronica e attraverso il Bus I²C e qualche modifica al programma facciamo in modo che sappia dove sta andando.

Non riesco a pensare che Aspirone vaghi per casa, svegliato solo dagli urti dei suoi baffi contro un mobile o una sedia, come ogni appassionato di robotica vivo il forte desiderio di fornire sempre più sensori alle mie creature, in fondo la differenza tra una macchina e un robot nel nostro immaginario è ben chiara: il robot ha cognizione di dov'è e di cosa fa, la macchina no, è solo sapendo e conoscendo l'ambiente in cui si opera che un robot può decidere cosa fare.
Vagando per la rete mi ero più volte imbattuto in schede che rilevassero il campo magnetico terrestre e alla fine ho ceduto, acquistata.
La scheda a cui mi riferisco è la CMPS03 basata su un integrato della Philips sensibili al campo magnetico terrestre, con due di questi integrati, il circuito fornisce attraverso il protocollo I²C un valore che può essere compreso tra 0 e 255 oppure 0 e 3599, tale valore è proporzionale ai gradi rispetto al polo nord magnetico.
La scheda reperibile in rete sia in Italia che oltre oceano viene fornita montata e collaudata come la vedete in figura 1, è in tecnologia SMD e le dimensioni sono veramente ridotte, solo 3x3 cm, il lavoro da fare non è molto, si tratta di trovargli una collocazione adeguata e collegare opportunamente i 6 pin necessari al funzionamento.

Posizionarla in realtà si è rivelata cosa ardua, tenendo conto della sensibilità ai campi magnetici, sarebbe piuttosto intelligente tenerla lontana dai motori.
La posizione migliore che ho trovato è stata di fissarla sopra la camera di espansione del circuito aria sfruttando una vite già presente nel plexiglas, una valida alternativa era di appoggiarla sopra la batteria con un po' di biadesivo. In figura 2 vedete la sistemazione che ho scelto

Diciamo subito che non è importante la direzione in cui la montate, ne che sia perpendicolare al robot, dovremo

Figura 1: *La scheda CMPS03, viene fornita montata e collaudata*

Figura 2: *Fissaggio della bussola sopra il circuito aria*

comunque gestire dei riferimenti relativi e matematici, per noi è più facile capire che 180 gradi significa Sud, per il robot 180, 128, 32768 o 430 sono la stessa cosa se gli diciamo che quello è il Sud.
Passiamo al collegamento, in tabella 1 vedete riportata la denominazione dei pin.

Tralasciando I pin non collegati, il modulo necessita ovviamente di una alimentazione (pin 1 e 9), il pin 7 dice al circuito qual è la frequenza di rete in modo da eliminarne i disturbi, nel nostro caso 50Hz questo pin va collegato a massa, nel caso dei 60Hz può essere lasciato scollegato visto che è dotato di una resistenza di pull-up.
Il pin 6 serve per la calibrazione, la procedura è piuttosto semplice e non va ripetuta ad ogni accensione, procedete così: con il circuito alimentato e parallelo al pavimento ruotatelo con il lato opposto a quello dove sono i pin verso Nord, collegate a massa il pin 6, scollegatelo e ruotate la bussola verso Est, date un altro impulso al pin 6 come prima e ripetete in direzione Sud e quindi Ovest.
Finito, la bussola è calibrata, potete ovviamente ripetere la procedura se lo ritenete opportuno, nella puntata precedente vi avevo fatto notare un pulsante sulla basetta del circuito di alimentazione, serve appunto a questo scopo.
Per prelevare i dati dalla bussola abbiamo due possibilità, o il segnale PWM sul pin 4 o utilizzare i pin 2 e 3 per il protocollo I²C.
Il segnale PWM ha le seguenti caratteristiche:

0° = 1ms
359,9° = 36,99ms

In altre parole abbiamo 100 microsecondi per ogni grado più 1ms, il timer interno ha una risoluzione di 10 microsecondi per cui si possono apprezzare variazioni dell'ordine del decimo di grado; tra un impulso e il successivo il segnale va a zero per 65ms.

Pin 1	+ 5V	Alimentazione (5 Volt)
Pin 2	SCL	Segnale del protocollo I²C
Pin 3	SDA	Segnale del protocollo I²C
Pin 4	PWM	Uscita PWM del dato riferito ai gradi
Pin 5	Non collegato	
Pin 6	Calibrazione	Pin per la procedura di calibrazione della bussola
Pin 7	50/60Hz	Filtro per I disturbi della rete elettrica 50 o 60 Hertz
Pin 8	Non collegato	
Pin 9	0V	Alimentazione (massa)

Tabella 1: *Denominazione dei pin della scheda CMPS03*

Visto che la scheda montata su Aspirone è dotata del connettore per il bus I²C la cosa migliore è di utilizzarlo, per fare questo è sufficiente collegare i pin SCL e SDA della bussola agli omonimi della scheda del Mark.
Il protocollo prevede che ogni periferica collegata abbia un indirizzo, nel caso della CMPS03 questo è scritto nel firmware ed è fisso 0XC0, se avete esperienze con le memorie tipo 24C04 troverete che non ci sono differenze, diversamente eccovi un pezzo di codice in PICbasic che legge l'angolo e lo mette nella variabile BEAR:

```
'Dichiarazioni
'I2C SCL for slave
scl var PORTC.3
'I2C SDA for slave
sda var PORTC.4
'The control code for
'  the slave 11000000
ctrl con $C0
addr=$01
...
...
...
'SUB lettura bussola
readbussola:
I2CREAD
'read the data
sda,scl,ctrl,addr,[BEAR]
return
```

Come vedete il PICbasic prevede un comando per leggere direttamente il bus, è sufficiente passare al comando i parametri relativi alle porte SCL e SDA, e l'indirizzo per avere nella variabile indicata il valore.
Nell'esempio ho utilizzato il

valore in un solo byte (0-255).
Non serve altro, la bussola e il nostro robot, lo vedete completo in figura 3 sono pronti a nuove missioni.

Avrete sicuramente idee su come sfruttare questa nuova caratteristica, prima di farlo leggete il paragrafo che segue… non c'è rosa senza spine.

I PROBLEMI DEL CAMPO MAGNETICO TERRESTRE

Devo confessarvi che è stata una amara sorpresa vedere Aspirone apparentemente ubriaco vagare per casa con continui e inspiegabili cambiamenti di rotta, il programma era stato modificato in modo da leggere all'avvio la direzione nella quale veniva acceso e da questa procedere correggendo eventuali errori monitorando con continuità la bussola, a nulla era valso creare delle bande di tolleranza.
Durante una delle tante prove ho avuto la fortuna di far passare Aspirone davanti ad una delle casse dello stereo e, ovviamente, Aspirone è impazzito puntando la cassa, era chiaro, in casa il debole campo magnetico terrestre è influenzato da tantissimi oggetti (metallici).
Potete verificare questo effetto con una bussola tradizionale camminando in casa.
È chiaro che il robot non faceva altro che seguire queste perturbazioni, che fare?
La soluzione non è poi difficile, anche se non mi permette di gestire la correzione del moto.
Lascio andare avanti il robot e a questo punto il fatto che vada diritto è affidato solo alla taratura dei servo, rilevato un ostacolo con i baffi fermo il robot che a questo punto legge la bussola, ad esempio il valore 40, se sono nella condizione di dover ruotare di 180° inizierò la rotazione e leggendo la bussola fino ad arrivare a 40+128=168.
In questo modo infatti l'eventuale errore dovuto ad interferenze in quel punto della casa è costante, poco importa se il 40 iniziale è vero o sbagliato.
La bussola nasce per la navigazione in mare aperto e anche lì sono necessarie verifiche e compensazioni dovute alla struttura della nave, è facile pertanto capire come possa risultare inaffidabile entro le mura domestiche questo tipo di navigazione; detto questo però posso assicurarvi che, vedere il robot iniziare a ruotare e fermarsi o correggere fino ad ottenere i 180° desiderati, è estremamente confortante rispetto alla soluzione di tarare dei tempi che non tengono conto del livello di carica della batteria o dello sforzo dei motori, rischiando quindi errori grossolani.

Figura 3: *Aspirone nella sua versione definitiva*

Abbiamo aggiunto una conoscenza al nostro robot, per chi si è appassionato dell'argomento voglio svelare che Aspirone è nato quasi come prototipo di un progetto simile, il Giardiniere, un robot che doveva occuparsi di tagliare l'erba del prato, se ci pensate, la meccanica e i concetti visti insieme possono adattarsi bene e la nostra bussola diventa sicuramente più affidabile.

Organizza

I° Workshop sul Device Networking

In collaborazione con: **SENA TECHNOLOGIES**

Sessione Mattutina (09.00-13.00)

Embedded Internet

Il corso permetterà al partecipante di apprendere nozioni sulla configurazione ed integrazione dei dispositivi Sena Technologies (www.hellodevice.it) serie HD1x00, Serie Super SS110, SS400, SS800

Argomenti principali della sessione:

- problematiche di connessione di apparecchiature e dispositivi elettronici alle reti TCP/IP
- i micro web servers Sena Technologies ed i loro possibili campi di applicazione
- tecniche per il controllo remoto e via web
- domande e risposte

Quota di partecipazione € 99,00 + IVA*
compreso un micro web server
mod. HD1100 del valore di € 125,00 + IVA

Sessione Pomeridiana (14.00-18.00)

L'interfacciamento dei dispositivi seriali alle reti LAN

Il corso permetterà al partecipante di apprendere nozioni sulla configurazione ed integrazione dei dispositivi Sena Technologies (www.hellodevice.it) serie Lite, Pro, STS, VTS.

Argomenti principali della sessione:

- la conversione Seriale/Ethernet semplice ed economica
- bridge di dispositivi seriali mediante LAN
- interfacciamento di dispositivi con diversi livelli di integrazione (dal modulo all'apparato multiporta)
- il console management
- domande e risposte

Quota di partecipazione € 99,00 + IVA*
compreso un serial/ethernet converter
mod. LS100 del valore di € 125,00 + IVA

Iscriviti ad entrambe le sessioni al prezzo irripetibile di € 189,00 +Iva
compresi 2 sistemi del valore complessivo di € 250,00 +IVA (1 mod. HD1100 + 1 mod. LS100)

elettroshop ACADEMY

Scheda di preregistrazione

Da compilarsi in ogni sua parte ed inviare via fax al n. 02 66508225 o per e-mail ad academy@elettroshop.com
È possibile iscriversi online all'indirizzo www.elettroshop.com/academy

Evento di*: ❑ Roma ❑ Milano ❑ Padova ❑ Torino

Nome e Cognome ______ Azienda ______

Via ______ Cap ______ Città ______ Prov. ______

Tel. ______ Fax ______ E-mail ______

Sessione Workshop:

❑ Mattutina (€ 99,00 + Iva compreso un sistema HD1100)
❑ Pomeridiana (€ 99,00 + Iva compreso un sistema LS100)
❑ Entrambe (€ 189,00 + Iva compreso un sistema HD1100 e un LS100)

Firma

Autorizzo il trattamento dei dati personali ai sensi della Legge 675/96

* Seguiranno dettagli sulle date degli eventi (I Workshop si terranno nei mesi di Ottobre/Novembre/Dicembre 2004) e sulle modalità del saldo della quota di partecipazione

INTRODUZIONE ALLA ROBOTICA

quarta parte

CAPACITÀ SENSORIALE

di Massimilinao Bracci
m.bracci@farelettronica.com

"...Affinché ogni macchina subordinata a un ambiente esterno variabile possa funzionare efficacemente, è necessario sia fornita ad essa l'informazione relativa ai risultati della sua stessa azione, come parte dell'informazione in base alla quale essa deve continuare ad operare...Questo comando della macchina sulla base del suo funzionamento effettivo anziché del suo comportamento previsto è conosciuto come retroazione (feedback), e implica che i membri sensori, messi in azione dai membri motori, svolgano una funzione di rilevatori o segnalatori, cioè di elementi che indicano il comportamento..."
["Introduzione alla cibernetica", Norbert Wiener]

Sempre più simili a noi i robot, anche se non nelle sembianze ma almeno nelle capacità, interagiscono ancor di più con il nostro mondo.
Sia che si tratti di robot industriali, quindi robot complessi inseriti in un ambiente semplice e noto, oppure, di robot autonomi (umanoidi) cioè robot semplici inseriti in un ambiente complesso, il sistema robot può essere rappresentato mediante una schema logico semplificato (figura 1) dove si identificano tre sotto-sistemi: un sistema meccanico identificato mediante una carrozzeria (locomozione e manipolazione), un sistema sensoriale per interfacciarsi con l'esterno (e interno) e un sistema di governo conglomerante sia hardware che il software (linguaggio di programmazione, azione di controllo e diagnosi) il tutto mediato da una connessione ad anello chiuso (retroazione).

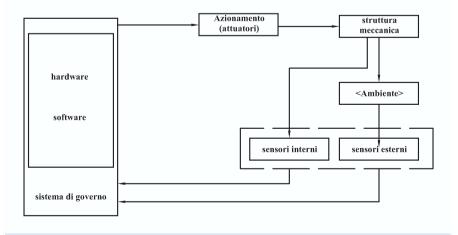



Figura 1: *Sistema robot - Schema logico semplificato*

I SENSORI

Le competenze su cui si basa la robotica sono di natura interdisciplinare:

- La *meccanica* dove si utilizzano le conoscenze della meccanica razionale e meccanica applicata per tutto quello che concerne la cinematica e la dinamica.
- La *elettrica* che interessa gli azionamenti dei motori elettrici e quelli idraulici o pneumatici.
- La competenza *elettronica* utilizzata per il controllo delle successive posizioni mediante i sensori.
- L'*informatica* necessaria per la programmazione della dinamica realizzata mediante un calcolatore, con l'ausilio di un opportuno linguaggio di programmazione.

Da questo punto di vista allora il sistema robot può essere rappresentato secondo uno schema a blocchi più dettagliato. (figura 2).

Figura 2: *Sistema robot - Schema logico più dettagliato*

Il sensore riceve dall'esterno un segnale di input che trasdotto viene elaborato dal calcolatore elettronico. In uno schema a blocchi si possono riassumere tutte le operazioni svolte, dall'acquisizione all'elaborazione dei dati (figura 3).

Figura 3: *Schema a blocchi delle varie operazioni svolte, dall'acquisizione all'elaborazione dei dati*

Il segnale acquisito dal sensore è di tipo analogico e di conseguenza per essere elaborato dal computer deve essere tradotto in digitale; risulta quindi necessario utilizzare dei convertitori A/D (Analogico/Digitale).
Una volta ricevuti ed interpretati i segnali digitali dal computer li si riconverte nuovamente in analogici per trasmettere il comando all'attuatore. Lo schema è quello di figura 4.

Figura 4: *Schema a blocchi: dal ricevimento del segnale alla trasmissione dei dati. Dall'attuatore e di nuovo al sensore tramite feedback*

Nel suo complesso la capacità sensoriale del robot può essere rappresentata mediante uno schema a blocchi come in figura 5. Mediante un sistema di controllo ad anello chiuso i valori delle grandezze in uscita vengono prima misurati e poi confrontati con quelli delle grandezze in ingresso. Il valore

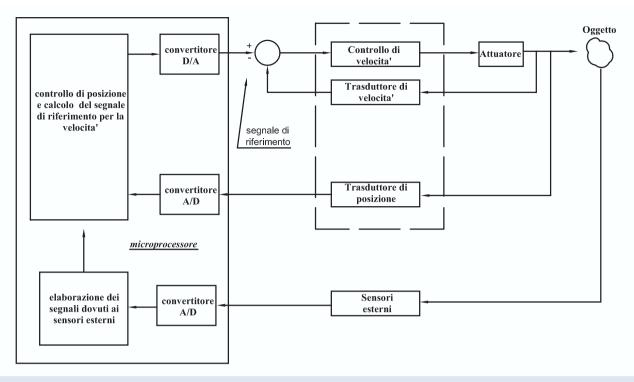



Figura 5: *Capacità sensoriale*

ottenuto da questo confronto, mediante un "segnale di riferimento", viene utilizzato come nuova grandezza in ingresso del sistema di controllo. In base a tale nuovo valore in ingresso, il sistema di controllo farà sì che la grandezza in uscita sia modificata fino a quando non raggiungerà un valore circa uguale a quello della grandezza di riferimento (entro determinati limiti prefissati in fase di progetto).

Gli elementi caratterizzanti un sensore si possono indicare come di seguito:

- *Rivelatore*: parte del sensore capace di captare il segnale interessato. La capacità di acquisizione del rilevatore è limitata dalla sensibilità dello stesso e dalla sua prontezza correlata all'intervallo di frequenze che lo strumento è in grado di apprezzare. Da tenere presente che sensibilità e prontezza vanno in direzione opposta l'una rispetto all'altra cioè l'elevata prontezza limita la sensibilità e viceversa.
- *Trasduttore*: parte del sensore che opera la conversione di una grandezza fisica di ingresso in una grandezza fisica di uscita di natura uguale o diversa da quella acquisita.
  Tutti i segnali di ingresso sono convertiti in segnali di tensione o corrente.
- *Sistema di acquisizione*: parte del sensore che è in grado di acquisire il segnale tradotto e renderlo leggibile per i passi successivi (usualmente il segnale tradotto è piuttosto debole così che necessita un amplificatore).
- *Sistema di elaborazione*: parte terminale del sensore che si occupa della elaborazione del segnale uscente. Nella maggior parte dei casi il segnale viene convertito da analogico in digitale.

I sensori più utilizzati nell'ambito della robotica sono i sensori di posizione, sensori di velocità, sensori di forza e pressione, di tatto, di distanza e sensori di visione.

SENSORI DI POSIZIONE E DI VELOCITÀ

L'encoder è un trasduttore di posizione e di velocità angolare che fornisce in uscita un segnale digitale. È un apparato elettromeccanico che converte la posizione angolare del suo asse rotante in un segnale elettrico digitale. I trasduttori sugli encoder impiegano un rilevamento

di tipo fotoelettrico. Gli encoder possono essere di due tipi: incrementali oppure assoluti.

Encoder incrementale

Segnala unicamente gli incrementi (variazioni) rilevabili rispetto a un'altra posizione assunta come riferimento. Questi incrementi sono indipendenti dal verso di rotazione. È sostanzialmente un disco di materiale plastico, sul quale sono stati ricavati dei fori o, più comunemente, alcune zone particolarmente trasparenti attraverso le quali è possibile il passaggio di un fascio luminoso. L'attraversamento del fascio luminoso nei fori comporta l'attivazione dell'uscita. I componenti che lo compongono quindi sono un disco, un fotoemettitore e un fotorilevatore (figura 6 e figura 7).

Figura 6: *Encoder incrementale*

Il disco è generalmente di plastica ed è calettato sull' albero dell' organo da controllare, diviso in zone chiare (trasparenti) e scure (opache).

Il fotoemettitore fornisce il segnale di input attraverso un segnale luminoso che passa nelle zone trasparenti del disco.

Il fotorilevatore riceve il segnale luminoso e a sua volta invia un segnale di output (logico 1 se passa la luce, logico 0 se non passa).

La rilevazione dello spostamento angolare avviene mediante il "conteggio" degli impulsi generati dal fotorilevatore alla propria uscita.Fissiamo l'attenzione su un disco costituito da N fori (figura 8).

La rilevazione dello spostamento angolare può essere così descritta: quando l'encoder si trova nella posizione di riferimento, il flusso luminoso attraversa il foro 0. In tal modo viene attivata l'uscita del fotorilevatore la quale si porta a livello alto, restandovi fino a quando il fascio luminoso viene interrotto. L'encoder inizia a ruotare e il fascio luminoso viene interrotto. Il fotorilevatore, quindi, risulta diseccitato e la sua uscita diventa bassa, fino a quando il foro 1 viene a trovarsi nella posizione occupata dal foro 0. Quando questo avviene il fascio luminoso attraversa il foro 1 eccitando nuovamente il fotorilevatore la cui uscita diviene nuovamente alta. Di conseguenza risulta noto lo sposta-

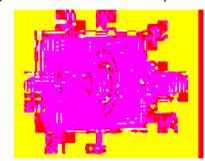
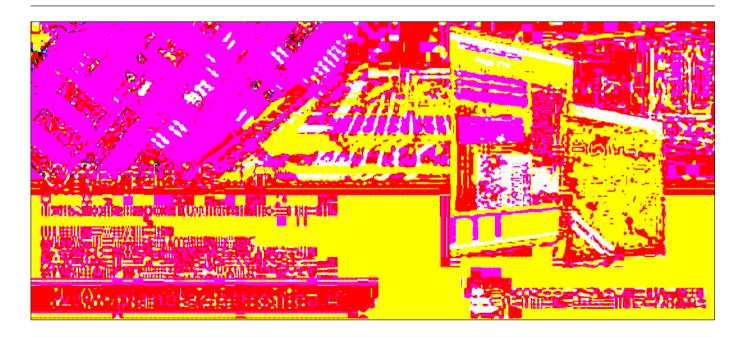



Figura 7: *Encoder incrementale nelle sue parti: (1) o-ring, (2) disco, (3) elettronica, (4) custodia, (5) testine di lettura, (6) cuscinetti, (7) albero.*

mento angolare 1 (=360° / (Numero di fori dell'encoder)).

Proseguendo nella rotazione il disco interrompe nuovamente il fascio luminoso diseccitando il fotorilevatore la cui uscita ritorna bassa.
Il processo descritto si ripete in modo perfettamente uguale nei passi successivi consentendo la rivelazione degli spostamenti angolari.
Il dispositivo digitale che rileva il numero di impulsi è un contatore, di conseguenza il numero degli impulsi contati è direttamente proporzionale allo spostamento angolare dell'encoder, cioè allo spostamento angolare dell'organo a cui è calettato.
L'encoder incrementale è adatto a rilevare rotazioni, velocità ed accelerazioni in base al conteggio degli impulsi inviati dal circuito in output.

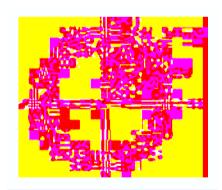



Figura 8: *Spostamento angolare*

Figura 9: *Codice binario – Codice Gray*

Encoder assoluto
Nel caso dell' encoder assoluto il disco è suddiviso in settori che andranno a comporre un codice (binario oppure *Gray*).
Il disco è diviso in n corone circolari e in 2 n spicchi. Ogni settore (spicchio) avrà n areole che a seconda se opacizzate o trasparenti corrisponderanno a 1 logico o 0 logico. Ogni areola avrà il valore di un bit.
Per evitare errori di lettura invece del codice binario puro vengono utilizzati altri codici, tra i quali il più importante è il codice Gray. Nel codice Gray il passaggio da un numero al successivo avviene sempre variando un'unica cifra binaria, evitando così che nel passaggio tra la lettura di un numero e del successivo possano aversi letture casuali. In questo caso ci sarà un fotoemettitore e un corrispondente fotorilevatore per ogni corona circolare del disco.

Gli encoder per la loro vastissima gamma di modelli, qualità e robustezza, sono validamente applicati in tutto il mondo su: controlli di processo industriale, robot industriali, macchine utensili, strumenti di misura, plotters, divisori, laminatoi, macchine per lamiera, bilance e bilici, antenne e telescopi, macchine per la lavorazione del vetro, marmo, cemento, legno, impianti ecologici, macchine tessili, conciarie, gru, carri ponte, presse, macchine da stampa, imballaggio, ecc.
In figura 10 vi sono riportati esempi di applicazione di encoder.

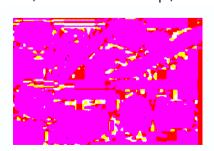



Figura 10: *Esempi di applicazioni di Encoder*

SENSORI DI FORZA E PRESSIONE

Questi sensori sono degli estensimetri, trasduttori capaci di eseguire la deformazione locale dell'elemento meccanico a cui sono rigidamente collegati. La deformazione viene poi legata agli sforzi tramite una relazione elastico-lineare di sforzi e deformazioni. Usualmente sono posizionati sui giunti delle pinze per misurare appunto la deformazione dei riscontri mobili durante il movimento e risalire inoltre anche alla forza applicata. (figura 11).

In modo analogo funzionano i sensori di pressione. Nella figura 11 vi è una schematizzazione di una pinza con dei sensori di pressione il cui posizionamento è solo su uno dei due riscontri.
Questi sono necessari per capire se l'oggetto in presa sta scivolando o meno. Si può constatare inoltre che i sensori di pressione sono due: uno è adibito alla misurazione della risultante normale e l'altro per misurare la risultante tangenziale, questo in seguito alla legge di Coulomb nota dalla meccanica razionale.

SENSORI TATTILI

Il sensore del tatto può essere imitato utilizzando sia dei sensori in cui occorre il contatto fisico tra sensore ed oggetto, banal-

mente dei fine-corsa a contatto meccanico utilizzati ad esempio per verificare la posizione tramite una grandezza off-on, oppure utilizzando dei sensori di prossimità. Questi ultimi, più sofisticati, sono sensori utilizzati per controllare la presenza di un oggetto o di un ostacolo in un intervallo di distanze specificato e comunque entro distanze brevi. Possono essere principalmente suddivisi nei seguenti tipi: sensori induttivi, sensori ad effetto Hall e sensori capacitivi.
Gli induttivi sono utilizzati per rilevare la presenza di oggetti ferromagnetici dotati di movimento relativo rispetto al sensore. La presenza di un oggetto di materiale ferromagnetico nelle immediate vicinanze del sensore provoca un variazione delle linee di flusso del campo magnetico generato dal magnete permanente.
Tale variazione viene sentita in modo dinamico, cioè durante lo spostamento relativo oggetto-sensore, come un impulso della corrente indotta nella bobina, con ampiezza proporzionale alla velocità di variazione del flusso. Il sensore e' efficace solo a brevi distanze (dell'ordine di qualche millimetro).
I sensori ad effetto Hall solo utilizzati per rilevare la presenza di materiali ferromagnetici e non necessitano di un moto relativo di questi. I sensori ad effetto Hall si servono di particolari componenti elettronici a semiconduttore sensibili ad un campo magnetico. Il principio di funzionamento si basa sul noto effetto Hall che di seguito viene brevemente richiamato.

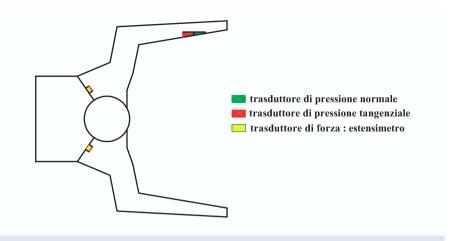



Figura 11: *Schematizzazione dei sensori in una pinza*

In un conduttore sagomato a forma di lamina percorso da corrente e soggetto ad un campo magnetico, si origina una differenza di potenziale in una direzione perpendicolare sia alla corrente che al campo magnetico. Il valore di tale differenza di potenziale è proporzionale all'intensità di corrente e all'intensità del campo magnetico, mentre è inversamente proporzionale allo spessore della lamina. Questo sensore in assenza di un oggetto da rivelare capta un forte campo magnetico emesso dal magnete permanente, mentre capta un campo più debole quando un oggetto ferromagnetico si trova nelle immediate vicinanze e varia in questo modo la conformazione delle linee di forza.
I sensori capacitivi possono rilevare oggetti di qualsiasi materiale, fermi o in movimento rispetto al sensore. Si basano sulla variazione di capacità generata dalla prossimità di un oggetto al sensore. L'elemento sensibile e' un condensatore formato da un elettrodo a forma di disco ed un elettrodo di riferimento di forma anulare. Il resto del sensore è costituito da un circuito elettronico che ha il compito di rilevare le variazioni di capacità generate dalla vicinanza di un oggetto, per esempio mediante circuiti oscillanti la cui frequenza varia con la capacità. Anche in questo caso la sensibilità massima del sensore si ha nell'ambito di pochi millimetri e varia con il variare del materiale dell'oggetto rilevato.

SENSORI DI DISTANZA

Questi sensori hanno prestazioni superiori a quelli di prossimità; sono in grado di fornire la distanza di un oggetto con una superiore precisione di quelle appena trattati.
Esistono essenzialmente due metodi per la realizzazione di questi sensori: il primo si basa sull'utilizzo di ultrasuoni, mentre il secondo fa uso di una sorgente a luce laser.
In un sensore a ultrasuoni l'elemento principale è un trasduttore elettroacustico, piezoelettrico, o ceramico. Misurando con particolari circuiti elettronici il tempo intercorrente tra l'in-

vio di un impulso e la ricezione dell'eco, il sensore può fornire non solo informazioni sulla presenza di un oggetto ma anche sulla sua distanza. Oltre al metodo precedentemente descritto, esistono altri due sistemi basati rispettivamente sull'emissione di impulsi di luce laser e sull'emissione di un fascio continuo con slittamento di fase.
Nel primo caso la misura della distanza avviene misurando il tempo che intercorre tra l'emissione di un impulso di luce laser e il suo ritorno dopo che questa è stata riflessa dalla superficie dell'oggetto di cui si intende misurare la distanza. Questi sistemi necessitano campionature della misura in intervalli di tempo dell'ordine dei pico secondi.
Il secondo metodo si basa viceversa sull'emissione di un fascio laser continuo. Questo fascio viene diviso da una superficie semiriflettente in due fasci: un fascio di riferimento, che va immediatamente a colpire un sistema per la misurazione della fase; un fascio di misura, che va a colpire l'oggetto, torna indietro e arriva anch'esso al misuratore di fase.
Misurando lo sfasamento tra i due fasci è possibile risalire alla distanza dell'oggetto.

SENSORI DI VISIONE

I sensori di visione hanno lo scopo di identificare la posizione e l'orientamento dell'oggetto che si trova nel campo del sensore, al fine di guidare il robot verso l'oggetto. Essi sono impiegati anche per riconoscere oggetti sia per forma che per colore. La struttura tipica di un sistema di visione è illustrata nella figura 12.

Il compito di riprendere le immagini è affidato a una telecamera a stato solido. Il rilevamento avviene mediante un reticolo di elementi fotosensibili su un chip CCD (Couplet Charge Device) dove su ciascun elemento si produce una tensione proporzionale all'intensità di luce che lo colpisce, quindi proporzionale alla luminosità del corrispondente punto della scena ripresa. Dopo una conversione analogico/digitale i valori della tensione associati a ciascun elemento (pixel) proporzionali al suo livello di grigio, vengono memorizzati sotto forma di matrice numerica in un buffer di memoria RAM.
Successivamente la matrice numerica viene elaborata per evidenziare il profilo degli oggetti rispetto allo sfondo della scena e confrontati con quelli precedentemente memorizzati.
In base al profilo rilevato il sistema di visione determina la posizione dell'oggetto e il suo orientamento. Queste informazioni vengono successivamente trasmesse al controllo del robot.
Da tenere presente che l'occhio umano può distinguere una

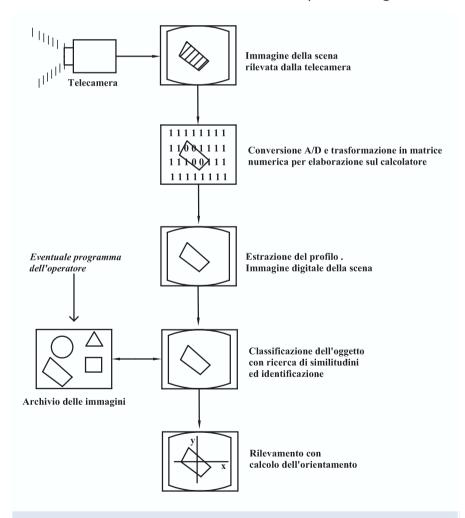



Figura 12: *Schema funzionale di un sistema di visione*

trentina di livelli di grigio mentre i sistemi di visione operano su almeno sessantaquattro livelli.

La procedura di ricerca del profilo degli oggetti consiste normalmente nella scansione della matrice numerica in cui sono stati memorizzati i valori di grigio di ciascun elemento di immagine. Per eseguire tale operazione esistono vari algoritmi, tra questi quello basato su di un livello di soglia di grigio.
Tale algoritmo consiste nel fissare un valore soglia di grigio (figura 13), al di sopra del quale il pixel deve considerarsi parte dell'oggetto così da essere associato al valore "0". Se il pixel è parte dello sfondo allora ad esso viene associato il valore "1" (figura 14). La successiva elaborazione verrà effettuata sugli elementi di valore 1 e porterà l'identificazione degli oggetti presenti nella scena.

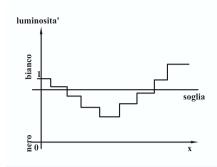
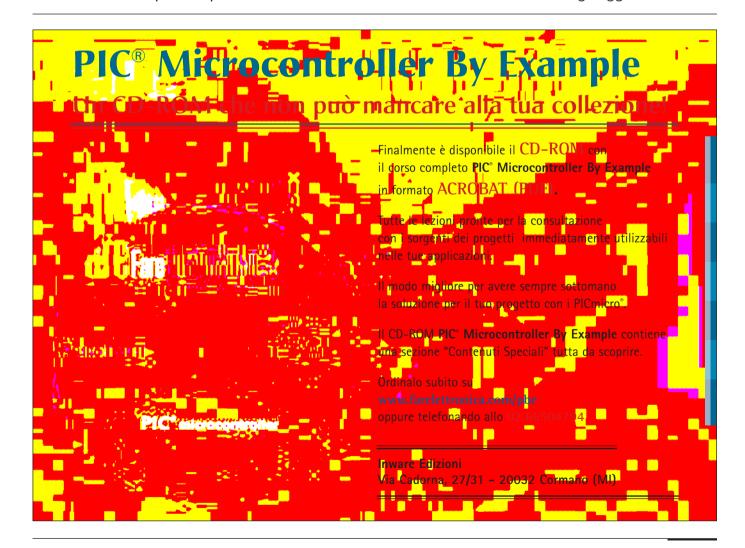



Figura 13: *Valore di soglia*

Figura 14: *Associazione del valore 0 oppure del valore 1 in funzione della luminosità rispetto al valore della soglia per il livello di grigio*

L'archivio delle immagini è una base dati residente nella memoria RAM del sistema, nella quale sono registrate alcune delle caratteristiche geometriche (come l'area oppure il perimetro…) tale che i loro valori garantiscano l'identificazione univoca degli oggetti.

Scheda di richiesta abbonamento

FE - 233

Si, desidero abbonarmi a Fare ELETTRONICA a partire dal primo numero raggiungibile

Cognome ______________________ Nome ______________________

Azienda ______________________

Via ______________________ CAP ________ Città ______________ Prov. ______

Tel. ______________ Fax ______________ email ______________________

Abbonamento: *(barrare la casella prescelta)*

❑ **Standard:** Mi abbono a Fare Elettronica per un anno (11 uscite) a soli € 39,00 anziché € 51,00

❑ **Rinnovo:** Sono già abbonato ed intendo rinnovare il mio abbonamento in scadenza. Fare Elettronica per un anno (11 uscite) a soli € 39,00 anziché € 51,00, il mio codice abbonamento è ______________

❑ **Regalo:** Regalo ad un amico Fare Elettronica per un anno (11 uscite) a soli € 35,00 anziché € 51,00
Riservato agli abbonati, il mio codice abbonamento è ______________

❑ **Scuole:** Cinque abbonamenti a Fare Elettronica per un anno (11 uscite) a soli € 156,00 anziché € 195,00
Riservato a Scuole ed Università

Pagherò con: *(barrare la casella prescelta)*

❑ **Bollettino postale** Utilizzare il **C/C N. 22790232** intestato ad **Inware srl**, indicando nella causale **"Abbonamento a Fare Elettronica"**

❑ **Bonifico bancario** Appoggiarlo su: **Poste Italiane**
CIN: Z - ABI: 07601 - CAB: 01600 - C/C: 000022790232 intestato ad **Inware srl**

❑ **Carta di credito** VISA MasterCard
Titolare: ______________________
Numero: ______________________ Scadenza: ____ / ____

Per completare l'attivazione dell'abbonamento, prego comunicare gli estremi (data e modalità prescelta) dell'avvenuto pagamento via telefono al numero (+39) 02.66504794 o via fax al numero (+39) 02.66508225

Firma ______________________

Privacy. Il trattamento dei dati, in forma automatizzata e con modalità strettamente connesse ai fini, con garanzia di riservatezza, è finalizzato all'invio del presente periodico allo scopo di informare ed aggiornare i lettori e gli operatori del settore elettronico sulle novità che il mercato propone. Potranno essere esercitati i diritti di cui all'articolo 13 della legge 675/96 (accesso, correzione, cancellazione, opposizione al trattamento, ecc.). Il titolare del trattamento dei dati è Inware srl con sede a Cormano (MI) in via Cadorna 27/31. Nel caso si tratti di copia omaggio a titolo promozionale si rende noto che i dati provengono da archivi pubblici. Resta inteso che le informazioni in ns. possesso non saranno in nessun caso cedute a terzi.

Abbonati subito!

- Compila il coupon e invialo via fax al numero 02.66508225
- Abbonati on-line: www.farelettronica.com
- Spedisci questo coupon in una busta chiusa a INWARE Edizioni Via Cadorna, 27/31 - 20032 Cormano (MI)
- Chiamaci al numero 02.66504794